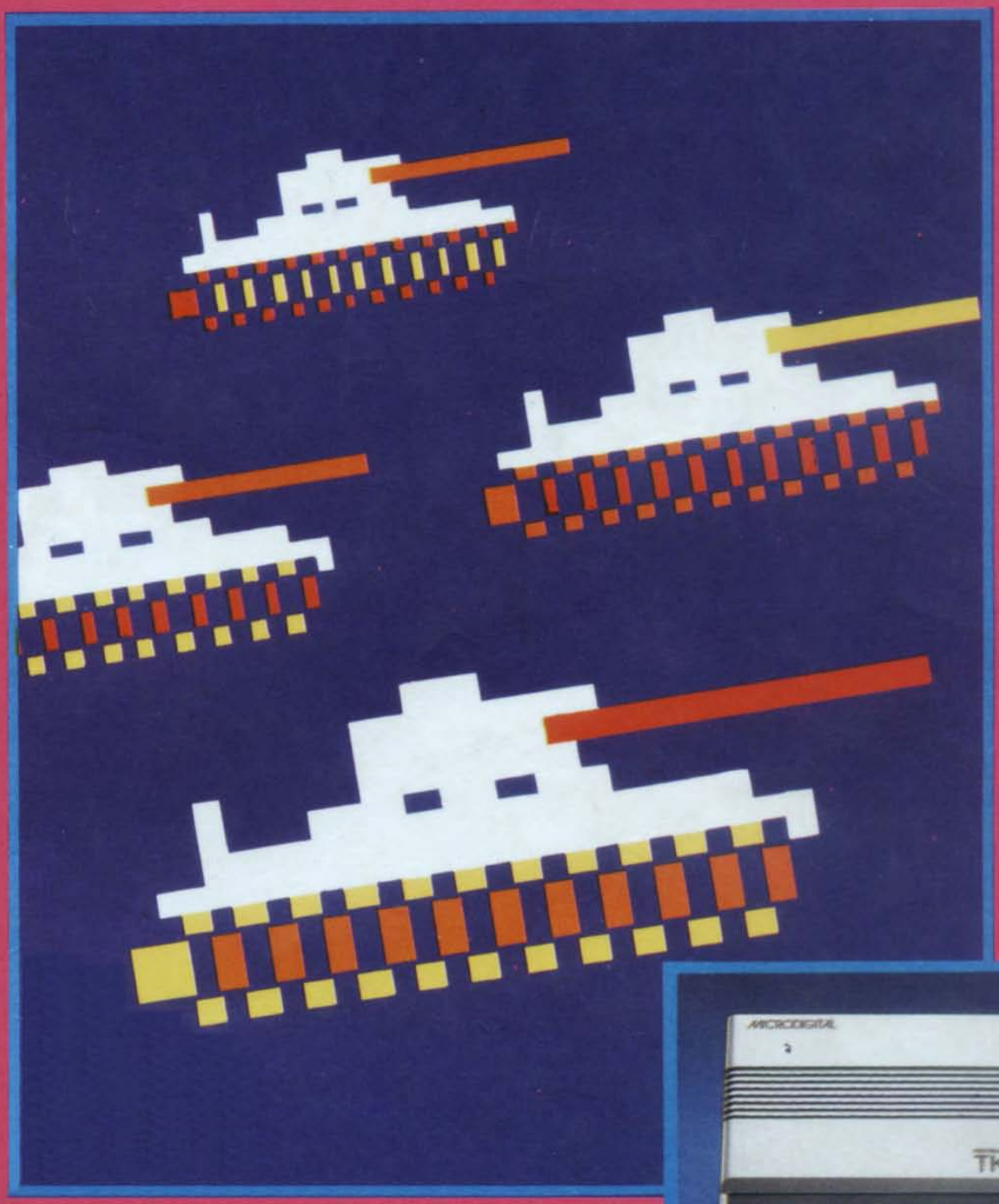
e-books

CURSO DE JOGOS EM BASIC TK

TK82, TK83 e TK85



micromega

EDITORA MODERNA

CURSO DE JOGOS EM BASIC TK

TK82, TK83 e TK85

FÁBIO RENDELUCCI

CURSO DE JOGOS EM BASIC TK

TK82, TK83 e TK85

1ª edição

EDITORA MODERNA

micromega

AGRADECIMENTOS A:
FLAVIO ROSSINI
RICARDO F. SIQUEIRA
CARLOS EDUARDO R. SALVATO

Com a popularização dos microcomputadores, observamos também uma verdadeira "fábrica" de jogos de vídeo.

A intenção deste livro é ensinar o funcionamento básico de um jogo, desde os simples até a um mais complexo, porém todos em BASIC, para computadores compatíveis com o TK83 (CP-200, RINGO, AS 1000, etc.).

Trataremos em cada capítulo de diversos "truques" utilizados em cada jogo, tais como placares, movimentos, níveis de dificuldade e até desenhos úteis, sempre acompanhados das devidas explicações e um jogo como exemplo, também explicado no seu diagrama de blocos.

Vocês verão que programar jogos não é tão complicado quanto parece, bastando apenas conhecermos a linguagem BASIC, portanto, mãos-à-obra e bom divertimento!

Fábio Rendelucci

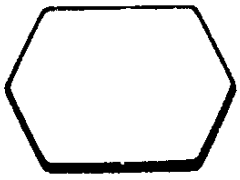
- *DIAGRAMA DE BLOCOS*

Antes de iniciarmos os jogos propriamente ditos, vamos apenas explicar como os esquematizaremos em sua idéia inicial, através do diagrama de blocos.

O diagrama de blocos serve para representarmos o "esqueleto" do programa, no caso o jogo, sem que tenhamos de escrevê-lo diretamente em BASIC. Os símbolos usados nos nossos diagramas serão os seguintes:

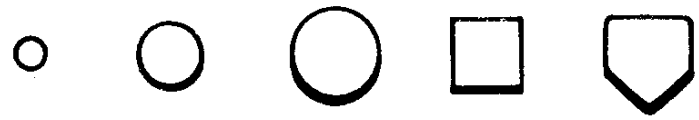
entrada manual

decisão

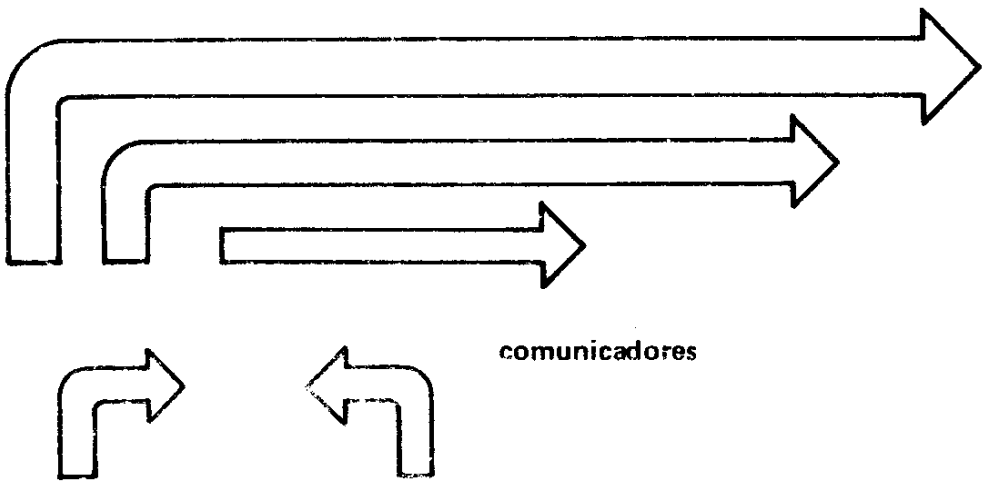


loop

Operação



conectores



comunicadores

Vejamos como os usaremos:

- *Operação*: Indicará qualquer função matemática e qualquer operação aritmética ou lógica, bem como geração de números aleatórios.

- *Início e fim*: Indica o começo e o fim do programa.

- *Vídeo*: Será usado quando devemos desenhar na tela, mostrar resultados, dados, etc...

- *Entrada Manual*: Será usado para indicar que forneceremos algum dado ao computador naquele instante.

- *Decisão*: Utilizado normalmente quanto temos que escolher um caminho, de acordo com determinadas condições.

- *Loop*: Indica o loop (FOR... NEXT). O NEXT será representado por um círculo que estará conectado no FOR. Colocaremos dentro dos blocos, apenas os limites do loop.
Daqui para frente, o diagrama de blocos será utilizado para que possamos compreender e elaborar os nossos jogos.

Como Elaborar Jogos

Certas vezes, quanto temos uma idéia para um jogo, temos alguma dificuldade em levá-la para frente, ou então nos complicamos em alguma parte e acabamos desistindo. Para evitar isso, devemos observar certas "regrinhas" na hora de elaborarmos nossos jogos:

1°) *Viabilidade*: verificar se a nossa idéia é realmente possível de ser colocada em prática, ou se não adianta nem tentar. Um dos maiores problemas será o tempo que o computador gasta no processamento. Não poderemos criar loops, principalmente que

envolvam movimentos muito longos, pois o tempo gasto no processamento não daria bom efeito no movimento. Outro problema muito comum é o tempo gasto no desenho da tela (o SLOW é muito lento e o FAST não fica bonito). Apesar de problemas deste tipo, podemos elaborar vários jogos, mas torna-se difícil implementar os jogos que apresentam várias telas ou movimento tridimensional.

2°) *Princípio de Funcionamento*: Depois de verificada sua viabilidade, devemos pensar como ele funcionará, o que regerá seus movimentos (se houver). Fazemos um esquema desse princípio e testamos. Não funcionando, reexamine a viabilidade do programa.

3°) *Diagrama de Blocos*: Depois de testado o princípio e tudo funcionando, pensamos nos complementos do nosso jogo (tempo, "combustível", placares, etc...) Fazemos então um diagrama total do jogo, observando agora todos os seus complementos.

4°) *Digitação*: Digitamos o programa com o mínimo de erros possíveis e testamos o programa. Caso apresente erro, verifique novamente o item 3.

5°) *Acertos Finais*: Tudo em ordem, basta agora darmos os últimos acertos, consertar os desenhos e gravar o programa.

JOGOS SIMPLES E JOGOS COMPLEXOS

Versões bem simplificadas de jogos são aquelas onde não há controle de tempo, de pontos e nem muitos movimentos seja de um alvo ou da sua peça. Jogos simples são portanto jogos que envolvem apenas um princípio de funcionamento.

Na verdade, jogos complexos são jogos simples, porém

"enfeitados" com controle de tempo, pontos, combustível, mensagens e telas elaboradas.

O jogo simples será, portanto, a estrutura do jogo complexo. No jogo complexo podemos ter então, vários jogos simples interligados de certa forma.

Neste capítulo iremos analisar dois jogos que utilizam o mesmo princípio: o lançamento oblíquo.

O primeiro deles é o Tiro ao Alvo. Neste jogo, devemos ter um alvo gerado aleatoriamente e um canhão fixo, que é controlado por nós. Após fornecermos um ângulo de tiro e uma velocidade inicial para o projétil, o canhão dispara e o nosso objetivo é atingir o alvo, que explodirá, caso tenhamos sucesso.

COMO FUNCIONA

A geração do alvo é feita por meio de dois números aleatórios que darão as coordenadas X e Y da posição do alvo na tela. O canhão é fixo na extremidade inferior esquerda da tela (origem dos eixos, para a função PLOT) e o tiro obedecerá à equação do movimento oblíquo, que conhecemos do nosso curso colegial:

$$\text{eixo x: } S = S\emptyset + V\emptyset x * t$$

$$\text{eixo y: } S = S\emptyset + V\emptyset * t + \frac{a * t^2}{2}$$

Onde:
S - espaço percorrido (posição final do objetivo)
SØ posição inicial do objeto
VØ velocidade inicial

a - aceleração (no caso, a aceleração da gravidade, g) os índices x e y em quaisquer variáveis indicam a que eixo se refere esta variável.

Forneceremos ao computador o ângulo de tiro (AN) em graus. O computador deve transformá-lo em radianos. A seguir, deveremos fornecer a velocidade inicial, . O computador calculará:

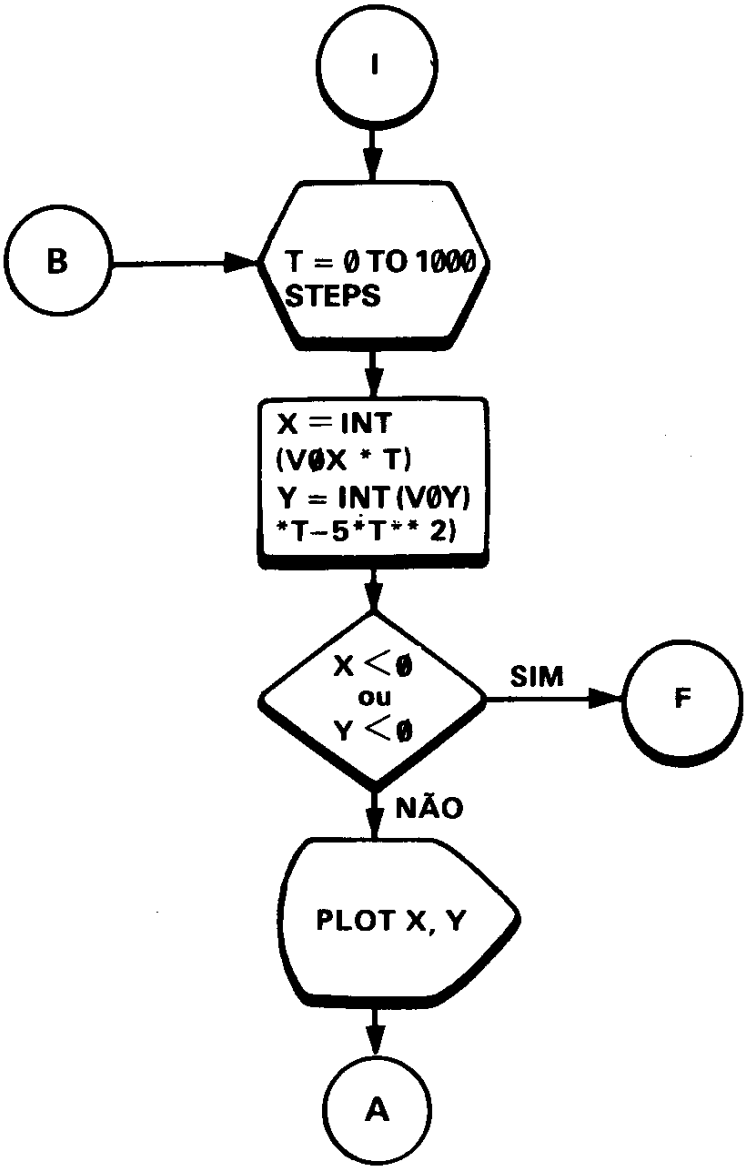
V0x = V0* COS AN
V0y = V0* SIN AN

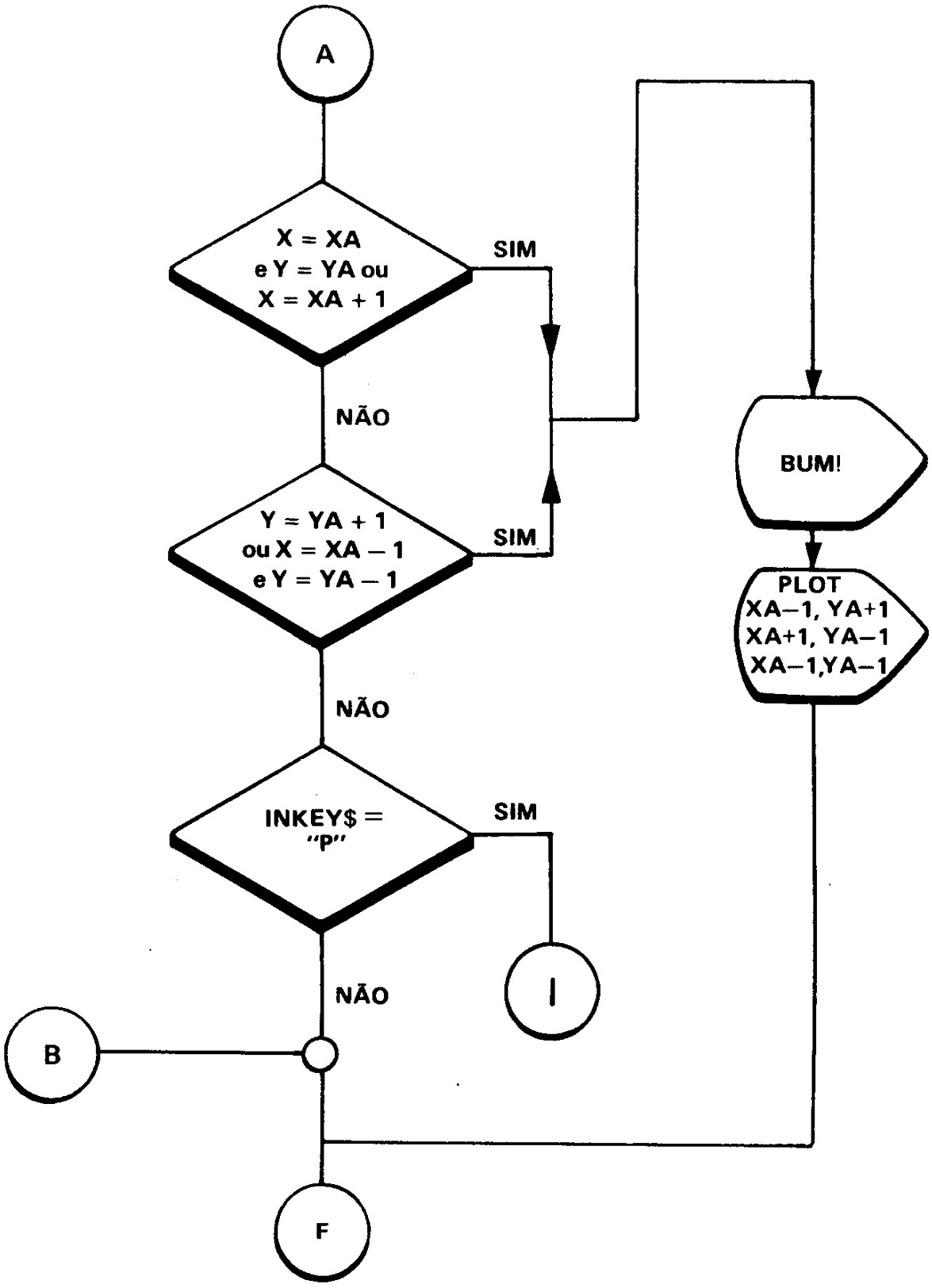
A aceleração "a" vale - 10(aceleração da gravidade), S0 é, para qualquer eixo, zero, pois o canhão está colocado na origem dos eixos x e y. O tempo, representado pela variável "t", será calculado pela variável contadora do *loop*, cujo passo vale 1/V0y ou 1/V0x.

Os PLOT usados são para os desenhos e a tecla P pára o programa, retornando ao início.

```
   5 REM TIRO AO ALVO
  10 LET XA=INT (RND*21+41)
  15 LET YA=INT (RND*30+2)
  20 CLS
  22 PRINT AT 1,6;"**TIRO AO ALV
O**"
  25 GOSUB 300
  30 PRINT AT 20,1;"V0=?"
  35 INPUT V0
  40 PRINT AT 20,1;"AN=?"
  45 INPUT AN
  50 LET AN=AN*PI/180
  55 PRINT AT 20,1;"     "
  60 LET V0X=V0*COS AN
  65 LET V0Y=V0*SIN AN
  70 LET S=1/V0X
  75 IF 1/V0X>1/V0Y THEN LET S=1
/V0Y
  80 FOR T=0 TO 1000 STEP S
  85 LET X=INT (V0X*T)
  90 LET Y=INT (V0Y*T-5*T**2)
  92 IF INKEY$="P" THEN GOTO 5
  95 IF X<0 OR Y<0 THEN GOTO 350
 100 PLOT X,Y
 105 IF (X=XA AND Y=YA) OR (X=XA
 AND Y=YA-1) OR (X=XA-1 AND Y=YA
-1) THEN GOTO 200
 110 IF X>62 OR Y>40 THEN GOTO 3
50
```

```
 115 NEXT T
 200 REM **EXPLOSAO**
 215 FOR A=-1 TO 1
 220 FOR B=-1 TO 1
 225 PLOT XA+A,YA+B
 230 NEXT B
 235 NEXT A
 240 CLS
 300 REM **DESENHA O ALVO**
 305 PLOT XA,YA
 310 PLOT XA-1,YA
 315 PLOT XA-1,YA-1
 320 PLOT XA,YA-1
 325 RETURN
 350 REM **FIM**
 355 PRINT AT 10,6;"F I M  D E
J O G O"
```



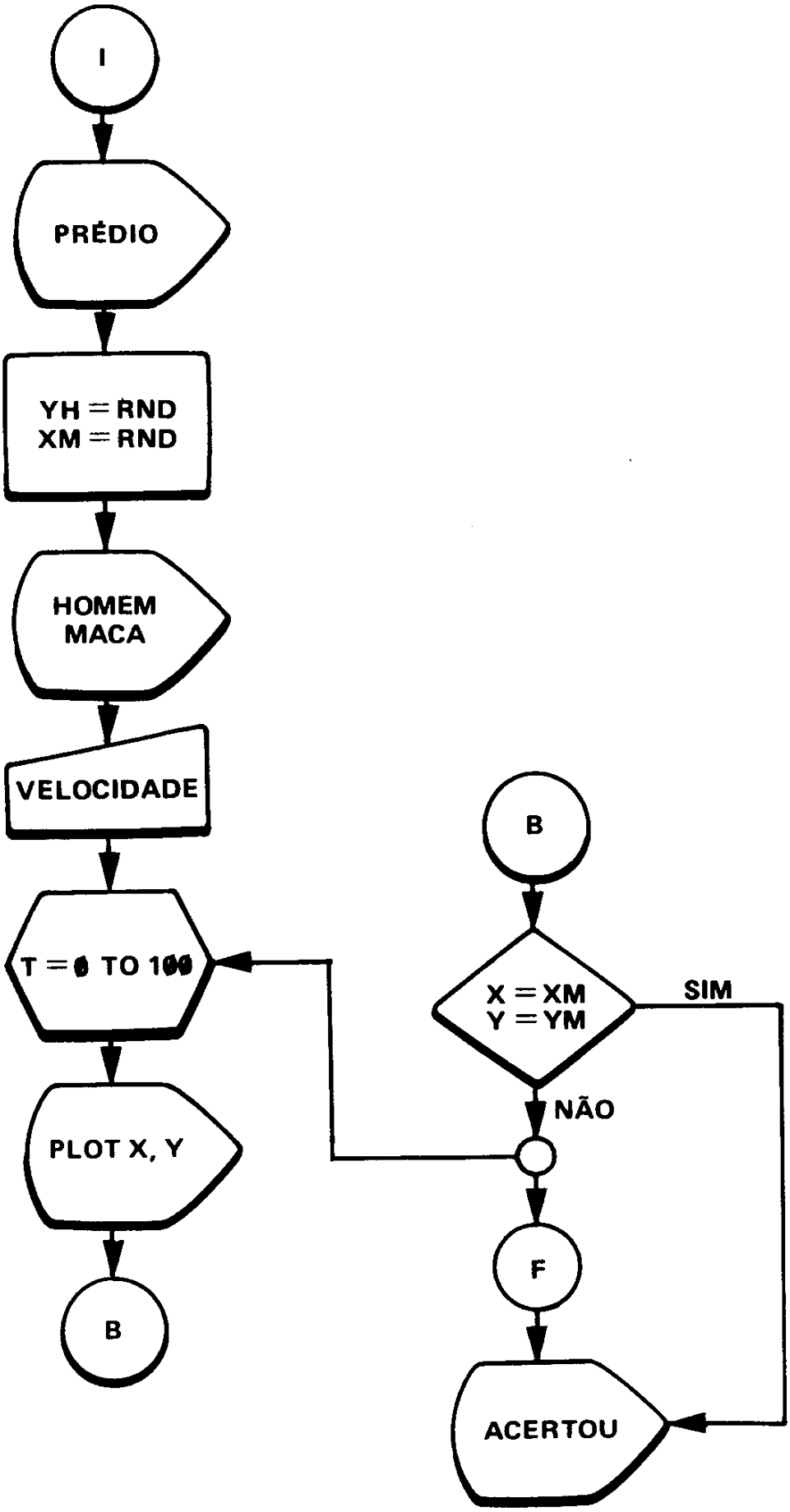

A
X = XA
e Y = YA ou
X = XA + 1
SIM
NÃO
Y = YA + 1
ou X = XA – 1
e Y = YA – 1
SIM
NÃO
INKEY$ =
"P"
SIM
NÃO
B
I
F
BUM!
PLOT
XA–1, YA+1
XA+1, YA–1
XA–1,YA–1

O funcionamento do "Salvamento" é muito parecido com o tiro ao alvo, apenas que o "tiro" agora será um "homem" que saltará de um prédio tentando cair numa maca. Agora forneceremos apenas a velocidade inicial.

O prédio é desenhado por dois loops e pontos PLOT. A posição da maca e o andar do prédio onde o homem se encontra são aleatórios.

```
   0 REM C-MICROMEGA 1983
   5 LET N=0
   7 LET M=0
   9 RAND
  10 PRINT TAB 6;"
";AT 2,0;"JOGO PARA 2 PESSOAS
";AT 4,0;"QUEM COMECA ?  (NOME)"
  20 INPUT N$
  25 IF LEN N$>7 THEN GOTO 20
  30 PRINT N$
  40 PRINT AT 7,0;"E O SEU OPONE
NTE ?"
  50 INPUT M$
  55 IF LEN M$>7 THEN GOTO 50
  60 PRINT M$
  70 PAUSE 90
  80 FAST
  90 CLS
 100 FOR X=1 TO 12 STEP 2
 110 FOR Y=0 TO 42
 120 PLOT X,Y
 130 NEXT Y
 140 NEXT X
 150 FOR X=2 TO 11 STEP 2
 160 FOR Y=0 TO 42 STEP 2
 170 PLOT X,Y
 180 NEXT Y
 190 NEXT X
 195 SLOW
 200 FOR W=0 TO 1
 210 LET H=INT ((RND*40)+2)
 220 LET D=INT ((RND*47)+14)
 225 PRINT AT 0,22;N;"    ";N$;AT
 1,22;M;"    ";M$;AT W,20;"";AT
3,25;"D=";D-12;" M";AT 4,25;"H="
;H;" M";AT 6,22;"V=???"
 230 PLOT 12,H
 231 PLOT D-1,0
 232 PLOT D,0
 233 PLOT D+1,0
 240 INPUT V
 250 PRINT AT 6,22;"V=";V;" M/S"
 260 FOR X=0 TO D-10
 270 LET Y=H-(5*X**2/V**2)
 280 LET R=X+12
 290 PLOT R,Y
 300 IF Y<=15 THEN GOTO 340
```

```
 310 NEXT X
 320 GOTO 600
 340 IF Y<=0 AND R<=D+1 AND R>=D
-1 THEN GOTO 500
 350 IF Y<=0 AND R<=D-2 THEN GOT
O 600
 360 GOTO 310
 500 IF W=0 THEN PRINT AT 10,7;M
$
 510 IF W=1 THEN PRINT AT 10,7;M
$
 511 IF W=1 THEN LET M=M+1
 512 IF W=0 THEN LET N=N+1
 520 FOR F=1 TO 15
 530 PRINT AT 10,16;"▒▒▒▒▒▒▒▒"
 540 PRINT AT 10,16;"ACERTOU"
 550 NEXT F
 560 FOR F=0 TO 21
 570 PRINT AT F,6;"
                  "
 580 NEXT F
 590 NEXT W
 595 GOTO 200
 600 IF W=0 THEN PRINT AT 10,7;N
$;" VOCE MORREU"
 610 IF W=1 THEN PRINT AT 10,7;M
$;" VOCE MORREU"
 620 PAUSE 100
 630 IF W=0 THEN LET M=M+1
 640 IF W=1 THEN LET N=N+1
 650 GOTO 560
 660 STOP
```


I
PRÉDIO
YH = RND
XM = RND
HOMEM
MACA
VELOCIDADE
T = 0 TO 100
PLOT X, Y
B
B
X = XM
Y = YM
SIM
NÃO
F
ACERTOU

Os efeitos aleatórios são um recurso muito utilizado nos jogos, pois eles constituem o fator sorte no *vídeo-game*. A geração de números aleatórios pode servir para gerar coordenadas para um alvo ou "inimigos", na colocação de obstáculos na tela.

Para a utilização de aleatórios, é bom lembrar:

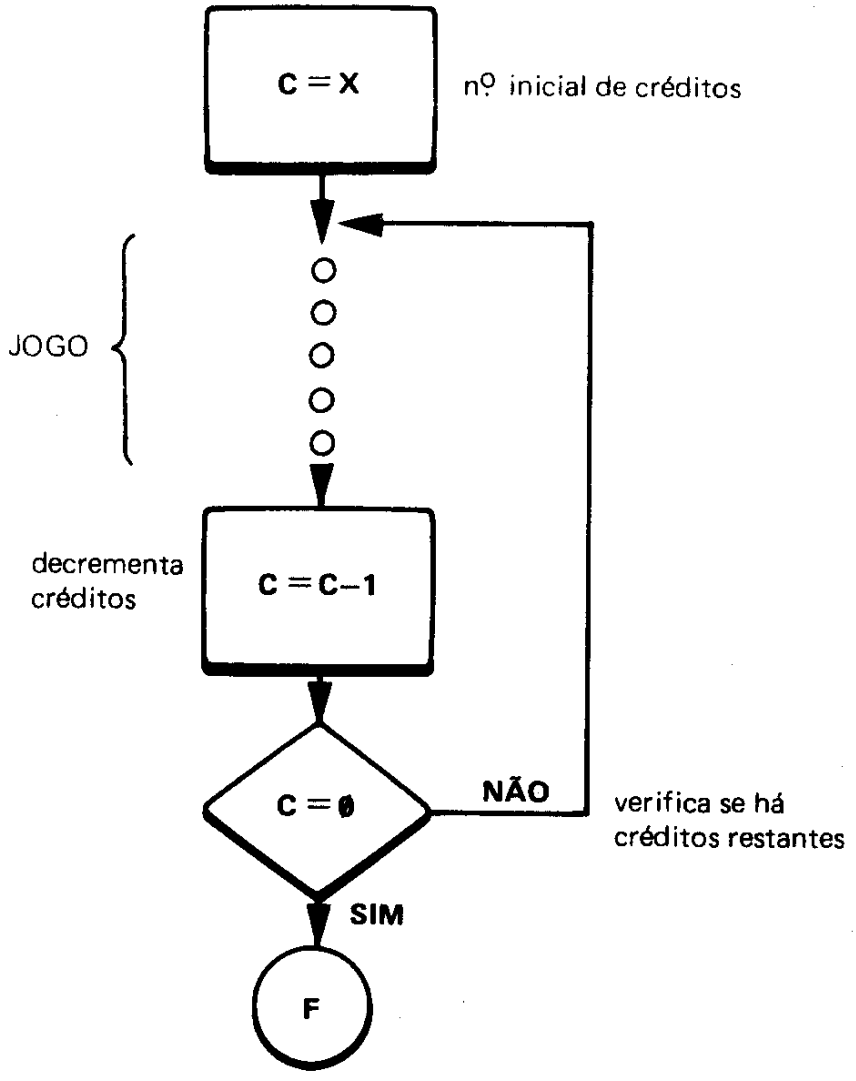
— coloque RAND no começo do programa. Isto torna os números verdadeiramente aleatórios.
— multiplicando-se o INT de um RND por "n", obtém-se um número entre Ø e (n-1).
— somando-se um número "m" ao INT (RND * n), obtém-se um intervalo entre m e m + n-1.

A utilização do aleatório num jogo pode representar a escolha aleatória de um evento ou de um conjunto de eventos possíveis no jogo.

Limite de Jogadas

É muito comum que, num jogo, tenhamos um limite de "chances" ou jogadas, para que não possamos jogar eternamente ou até ganharmos o jogo.

Podemos limitar o jogo de duas formas: ou o colocamos dentro de um *loop* ou criamos uma variável contadora, que funciona como "créditos" que possuímos, créditos estes que a cada jogada serão decrementados ou poderão ser acrescidos com um certo bônus.



INSTRUÇÕES

Notamos que em alguns tipos de jogos, temos várias teclas

para apertar, ou mesmo várias maneiras de proceder. Um recurso então se faz necessário: a impressão de instruções.

As instruções devem aparecer no início, de maneira clara e durante um tempo suficiente para sua leitura duas vezes.

CASSINO

Neste programa geramos caracteres aleatórios que aparecerão na tela em 3 quadrinhos. Um por um os caracteres são fixados (numa possibilidade de apenas 4 tipos diferentes). No final da geração, comparam-se os 3 para ver se são iguais, e atribui-se bônus aos caracteres iguais.

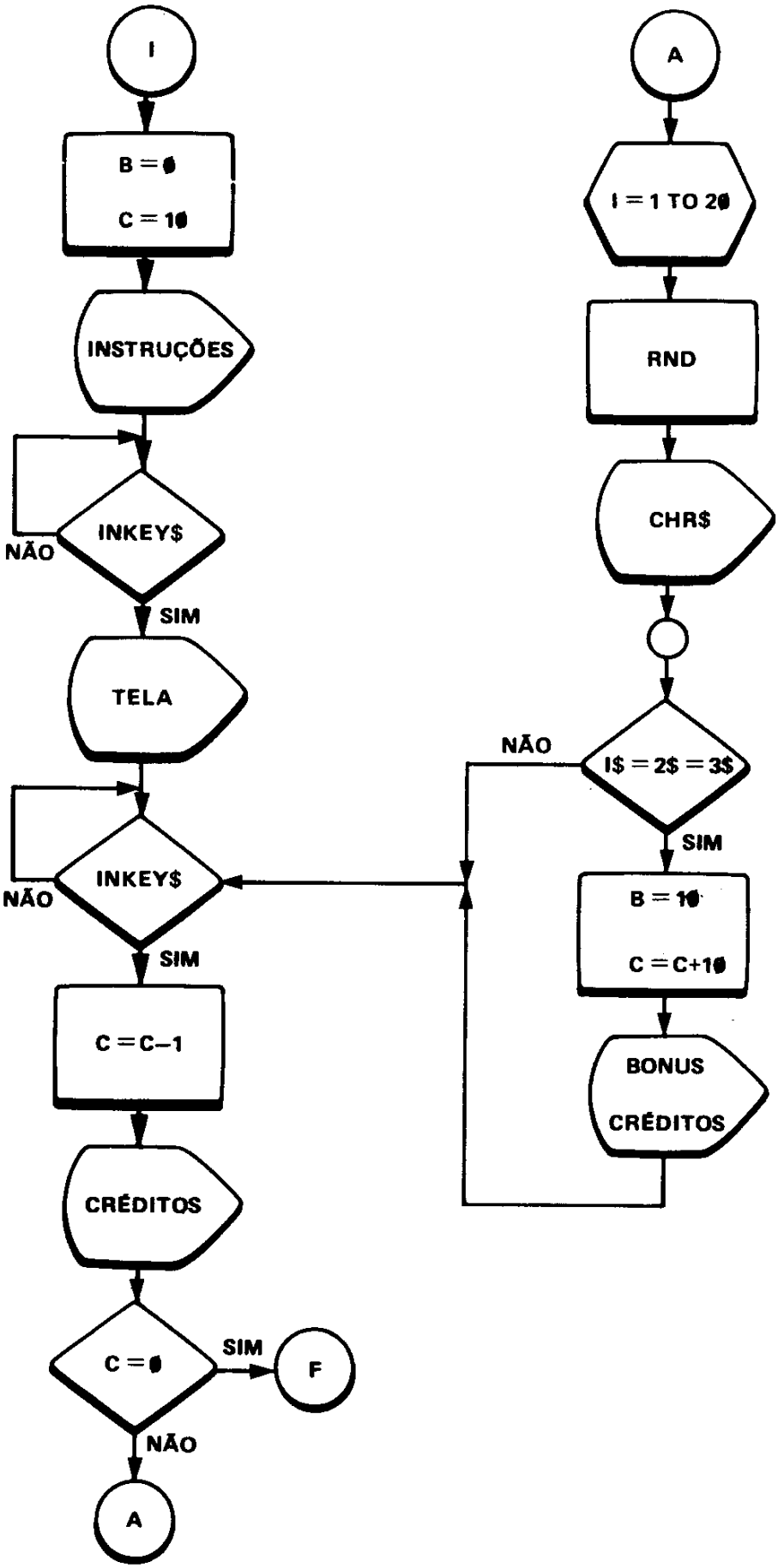
OBS.: Neste jogo, notamos a utilização das linhas 22 e 23 da tela. Isto só é possível com um comando

```
POKE 16418,0
```

anterior à impressão.

```
  0 REM C-MICROMEGA 1983
 63 POKE 16418,0
101 CLS
102 CLEAR
104 RAND
105 LET C=10
110 LET B=0
111 FOR G=0 TO 31
112 PRINT AT 0,G;"▒";AT 23,G;"▒
"
113 NEXT G
114 FOR H=0 TO 23
115 PRINT AT H,0;"▒";AT H,31;"▒
"
116 NEXT H
117 PRINT AT 2,8;"* CACA NIQUEI
S *"
118 PRINT TAB 12;"*"
120 FOR D=6 TO 8
125 FOR E=9 TO 11
130 PRINT AT D,E;"▒";AT D,E+4;"
▒";AT D,E+8;"▒"
135 NEXT E
140 NEXT D
145 PRINT AT 13,2;"> BONUS:";B
150 PRINT AT 16,2;"> CREDITO:";
C
```

```
  165 PAUSE 33000
  170 POKE 16437,255
  180 PRINT AT 10,9;">BOA SORTE<"
  185 FOR Y=0 TO 15
  190 NEXT Y
  195 PRINT AT 10,9;"                "
  205 LET C=C-1
  210 IF C=0 THEN GOTO 2000
  215 IF C<10 THEN PRINT AT 16,12
;"0";C
  220 IF NOT C<10 THEN PRINT AT 1
6,12;C
  225 LET G=10
  230 DIM J(3)
  232 FOR H=1 TO 3
  235 LET I=INT (RND*100)
  236 IF I<40 THEN LET J(H)=5
  237 IF I>=40 AND I<70 THEN LET
J(H)=134
  238 IF I>=70 AND I<90 THEN LET
J(H)=180
  239 IF I>=90 THEN LET J(H)=151
  240 FOR N=0 TO 15
  244 GOTO 243+(2*H)
  245 PRINT AT 7,G;CHR$ INT (RND*
11);AT 7,G+4;CHR$ INT (RND*11);A
T 7,G+8;CHR$ INT (RND*10)
  246 NEXT N
  247 PRINT AT 7,14;CHR$ INT (RND
*10);AT 7,18;CHR$ INT (RND*10)
  248 NEXT N
  249 PRINT AT 7,18;CHR$ INT (RND
*10)
  250 NEXT N
  255 PRINT AT 7,G;CHR$ (J(H))
  260 LET G=G+4
  265 NEXT H
  270 IF J(1)=J(2) AND J(2)=J(3)
THEN GOTO 290
  275 LET F=0
  280 LET C=C+F
  285 GOTO 165
  290 IF J(1)=5 THEN LET F=5
  295 IF J(1)=134 THEN LET F=10
  300 IF J(1)=180 THEN LET F=15
  305 IF J(1)=151 THEN LET F=20
  310 LET C=C+F
  311 PRINT AT 13,10;F
  312 PRINT AT 16,12;C
  313 FOR P=0 TO 5
  314 NEXT P
  315 PRINT AT 13,10;"00"
  316 GOTO 165
2000 PRINT AT 10,5;"FIM DE JOGO"
2005 FOR M=0 TO 30
2010 NEXT M
2015 CLS
2020 GOTO 1
3000 INPUT U$
3005 SLOW
3010 SAVE "SLOW"
3015 GOTO 1
```


I
B = 0
C = 10
INSTRUÇÕES
INKEY$
NÃO
SIM
TELA
INKEY$
NÃO
SIM
C = C–1
CRÉDITOS
C = 0
SIM
F
NÃO
A
A
I = 1 TO 20
RND
CHR$
I$ = 2$ = 3$
NÃO
SIM
B = 10
C = C+10
BONUS
CRÉDITOS

INCREMENTANDO PONTOS

Para incrementar o placar de um jogo, devemos primeiramente saber se os valores serão constantes ou não, isto é, se cada "tarefa" terá o mesmo valor, ou se esse valor varia de acordo com a dificuldade da "tarefa". Observe nos exemplos:

a) o computador vai gerar um nº aleatório quando você aperta uma tecla. Se o nº for maior que 5, você ganha 1 ponto.

```
 10 LET S=0
 20 RAND
 30 FOR A=0 TO 9
 40 PRINT "APERTE UMA TECLA"
 50 IF INKEY$="" THEN GOTO 50
 60 CLS
 70 LET N=INT (RND*11)
 80 IF N>5 THEN LET S=S+1
 90 PRINT AT 20,0;"PLACAR=";S
100 NEXT A
```

Note que para qualquer valor acima de 5 você ganha sempre 1 ponto.

b) Acrescente no programa anterior:

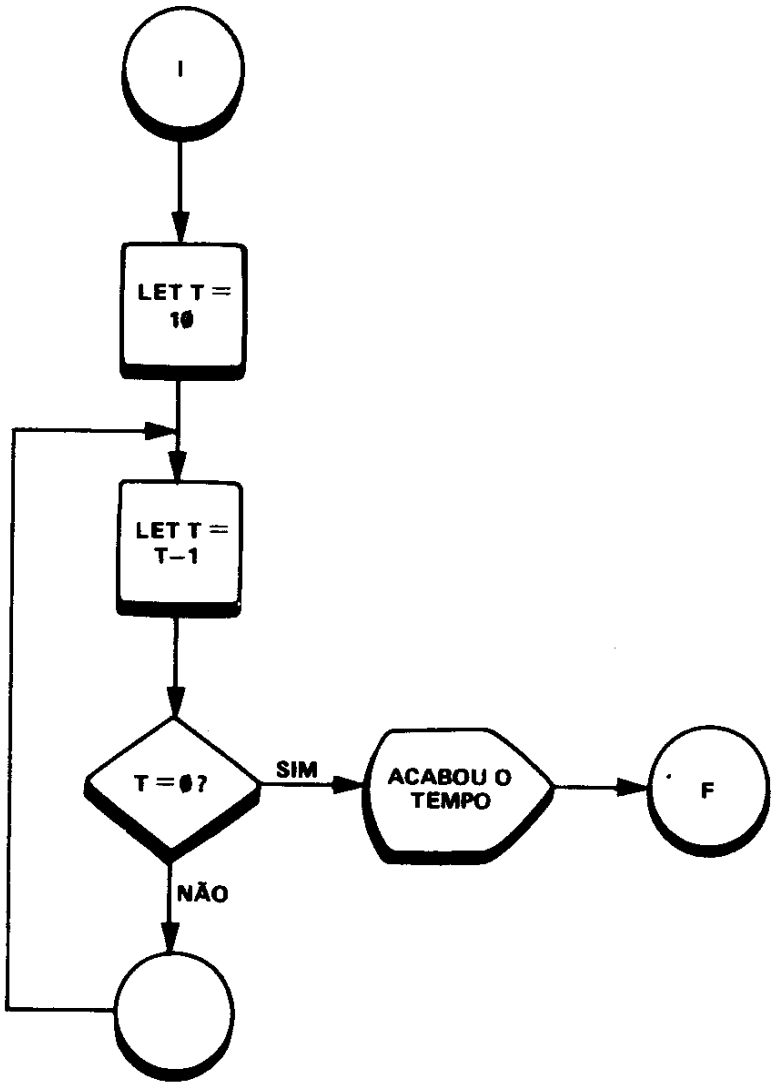
```
30 IF N>5 THEN LET S=S+N
```

Agora o computador gera um número. Se ele for maior que 5, você ganha tantos pontos quanto o valor do número sorteado e seus pontos variam de 6 a 10.

DECREMENTANDO TEMPO

Uma partida de *vídeo-game* costuma ter sua duração limitada por um destes fatores: combustível, número de naves ou tempo.

Controlar um jogo por tempo é bem simples.



Este é o princípio básico, porém devemos ter certos cuidados:

a) Se o loop do jogo for muito comprido, devemos decrementar o tempo mais de uma vez no loop.

b) Na impressão da tela, teremos um problema. Observe o exemplo:

```
10 LET T=110
20 PRINT AT 10,10;"TEMPO:";T
30 LET T=T-1
40 GOTO 20
```

Notamos que quando o tempo passa de 100 para 99, obtemos as seguintes mensagens:

```
TEMPO 100

TEMPO 990
```

O mesmo acontece de 10 para 9:

```
TEMPO 10

TEMPO 90
```

Isto ocorre porque o número 100 ocupa 3 caracteres e o 99 ocupa 2. Como não usamos CLS (para ficar mais bonito), o caracter 0 do final do 100 não será apagado.

Como fazer então?

Observe agora o exemplo:

```
10 LET T=110
20 PRINT AT 10,10;"TEMPO:";T
30 LET T=T-1
40 IF T<100 AND T>98 THEN GOTO
70
50 IF T<10 AND T>6 THEN GOTO 7
0
60 GOTO 20
70 PRINT AT 10,15;"   "
80 GOTO 60
```

Quando o tempo for 99 ou 9, o computador imprimirá três espaços brancos em cima do tempo anterior, apagando-o.

Colocamos T<100 AND>T 98 para que o computador só apague o anterior quando for necessário, não perdendo assim tempo real de execução.

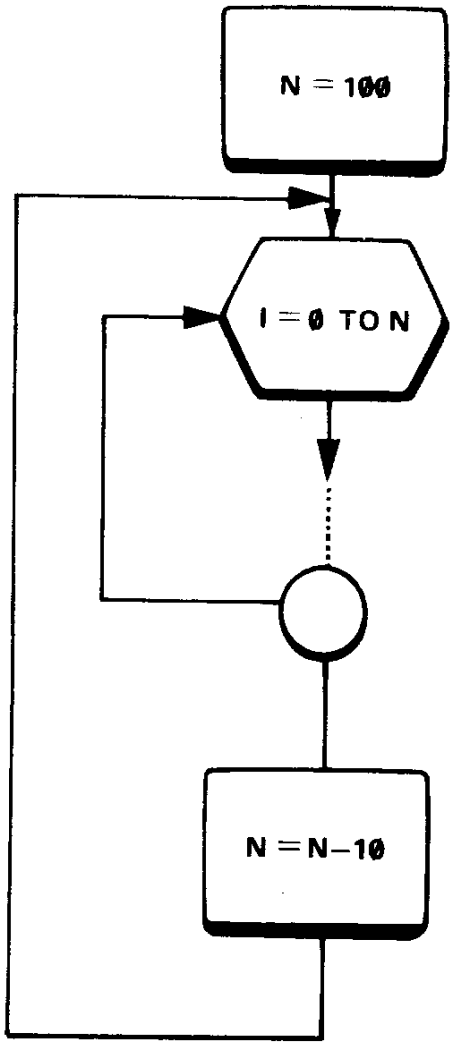
NÍVEL DE DIFICULDADE

A função do nível de dificuldade é, logicamente, complicar o jogo. Mas, como complicá-lo?

Isto vai depender muito do jogo. Por exemplo, no caso de jogos onde dependemos de números aleatórios, podemos, de acordo com o nível, diminuir a probabilidade de sorteio desses números.

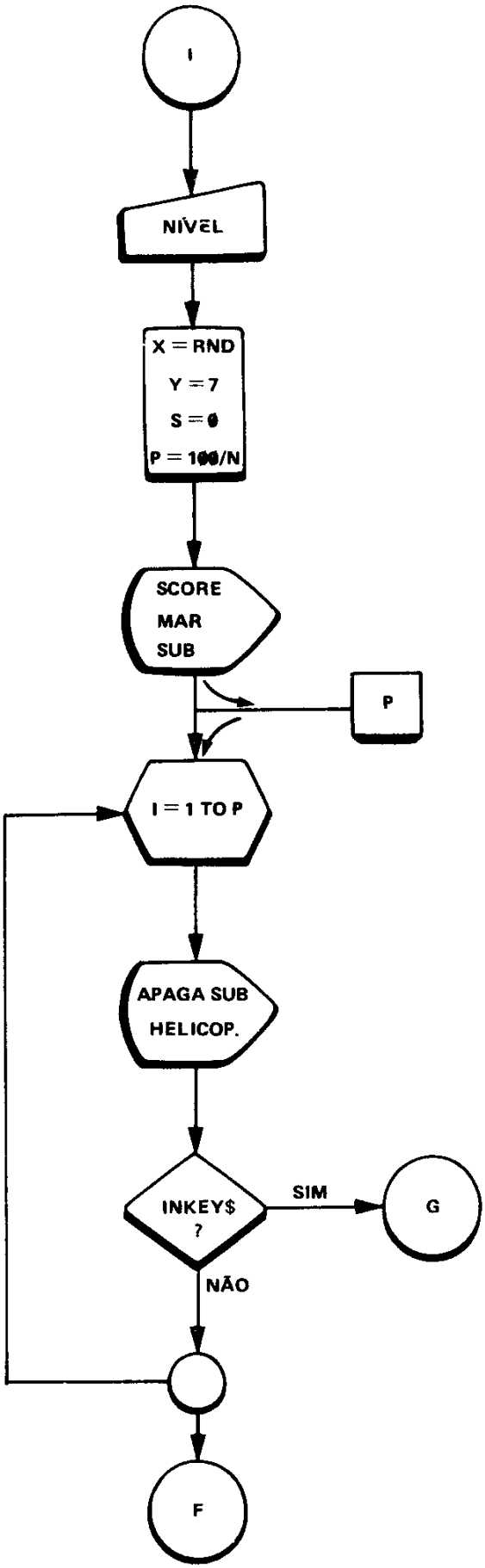
Outro artifício é diminuir o tempo do *loop*. Temos então menos tempo para realizar nossa "missão". Esse artifício é também usado em jogos onde o *loop* será repetido a cada vez que completamos um ciclo. Quando passamos para o ciclo seguinte, recomeçamos o jogo com um loop de tempo menor, tornando o jogo cada vez mais difícil.

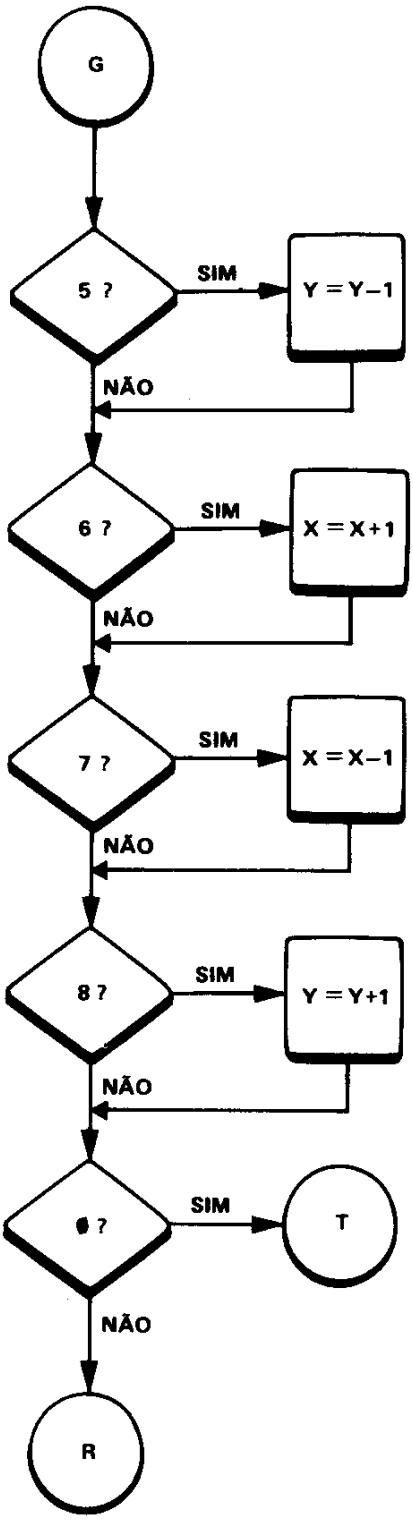
Observe o diagrama:



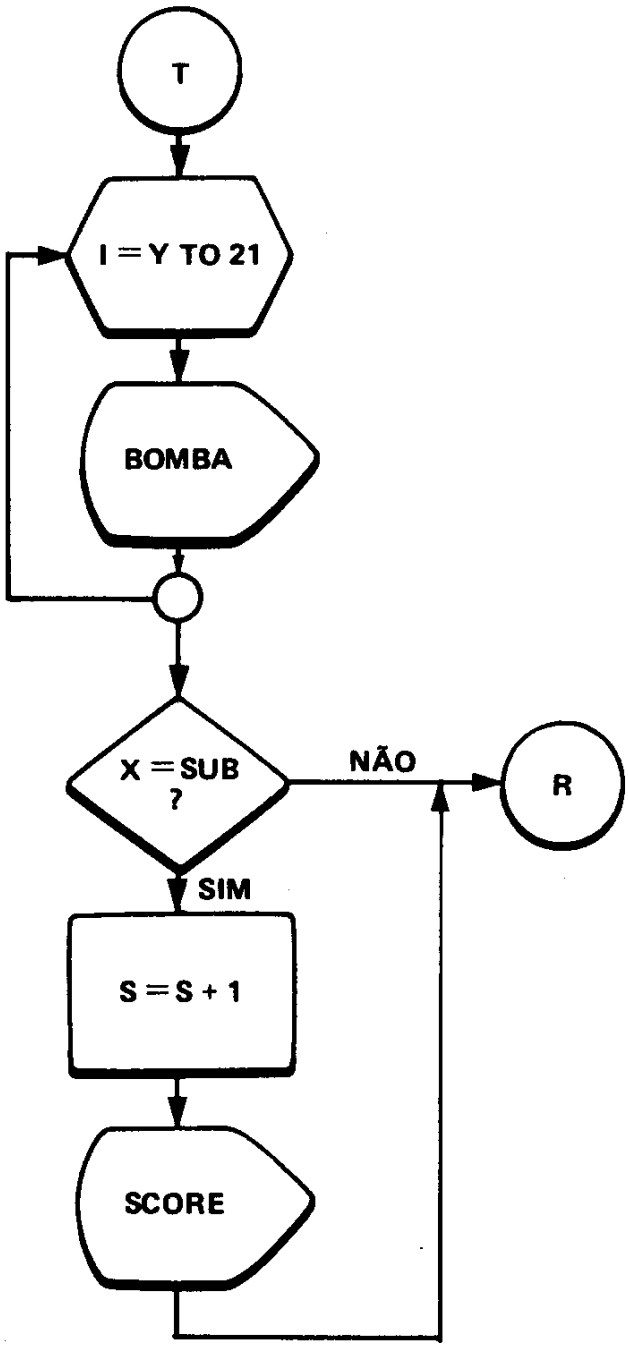
SUB-HUNTER

Neste jogo, pilotamos um helicóptero que deve bombardear um submarino que se torna visível (seu periscópio) por apenas alguns instantes (o tempo varia com o nível de dificuldade).


I
NÍVEL
X = RND
Y = 7
S = 0
P = 100/N
SCORE
MAR
SUB
P
I = 1 TO P
APAGA SUB
HELICOP.
INKEY$
?
SIM
G
NÃO
F


G
5 ?
SIM
Y = Y–1
NÃO
6 ?
SIM
X = X+1
NÃO
7 ?
SIM
X = X–1
NÃO
8 ?
SIM
Y = Y+1
NÃO
0 ?
SIM
T
NÃO
R

P = PAUSE




```
 1 RAND
 2 LET D=0
 5 SLOW
10 GOSUB 100
15 LET S=0
16 LET XH=10
17 LET YH=6
20 GOSUB 120
23 LET T=100-(10*N)
25 LET X=5+INT (RND*21)
30 LET Y=INT (RND*5)+17
35 PRINT AT Y,X;"■"
40 FOR I=0 TO (20/N)
```

```
  45 NEXT I
  50 PRINT AT Y,X;"/"
  55 FOR C=0 TO T-S
  60 PRINT AT YH,XH;"  ▄▄▄▄▄▄▄▄  ";
AT YH+1,XH;"  ▄▄▄▄▄▄▄▄0 ";AT YH+2,X
H;"   ▄▄▄▄   "
  65 LET XH=XH+(1 AND INKEY$="6"
)-(1 AND INKEY$="5")
  70 IF INKEY$="0" THEN GOTO 150
  75 NEXT C
  77 LET D=D+1
  80 IF NOT D=10 THEN GOTO 23
  85 PRINT AT 10,10;"FIM DE JOGO
"
  87 PRINT AT 12,6;"SEUS PONTOS:
";S
  90 STOP
 100 FOR A=21 TO 16 STEP -1
 105 PRINT AT A,0;"////////////////
////////////////////////"
 110 NEXT A
 115 RETURN
 120 PRINT AT 0,0;"GRAU DE DIFIC
ULDADE ? (1/2/3)"
 125 INPUT N
 130 PRINT AT 0,0;"
                  "
 135 RETURN
 150 FOR R=YH+4 TO 21
 155 PRINT AT R,XH+3;"*"
 156 PRINT AT R,XH+3;" "
 160 IF R=Y AND XH+3=X THEN GOTO
 200
 165 NEXT R
 170 GOSUB 100
 175 GOTO 77
 200 PRINT AT Y,X-2;"*****";AT Y
-1,X-1;"***"
 210 FOR W=0 TO 30
 215 NEXT W
 220 LET S=S+10
 225 GOTO 170
```

Um dos maiores problemas que encontramos na movimentação de nossas "peças", é apagá-las da posição anterior.

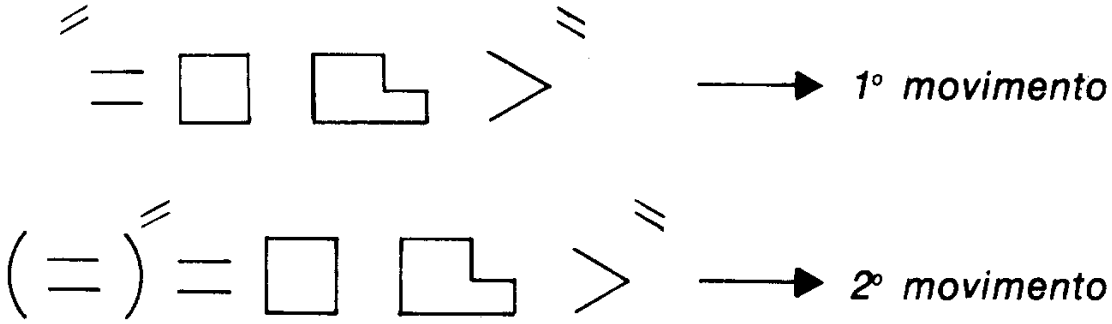
Se usarmos um CLS, apagamos a tela inteira, o que não pode acontecer. Se usamos a técnica de imprimir espaços na naves anterior, teremos o efeito de vê-la piscando, observe:

```
10 FOR I=0 TO 28
20 PRINT AT 10,I;"=■>"
30 PRINT AT 10,I;"    "
40 NEXT I
```

O efeito não é nada bonito.

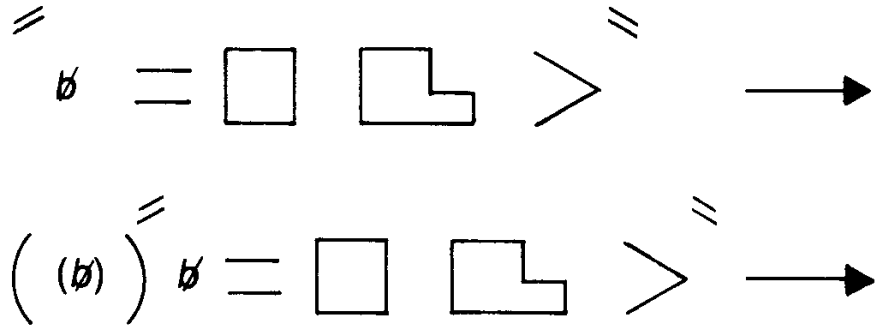
Usaremos então o seguinte artifício: sendo o movimento da esquerda para a direita, colocamos como primeiro caracter da nave, um espaço branco. Como o problema era apagar o primeiro caracter do desenho anterior, se ele for um espaço, não precisaremos apagá-lo.

Observe este esquema:



1º caractere ainda impresso

Veja agora:



1º caractere anterior (espaço)

Teste o programa

```
10 FOR I=0 TO 27
20 PRINT AT 10,I;" =";
30 NEXT I
```

O efeito agora é perfeito.

Outro artifício, para "naves" controladas por INKEYS, e fazê-las ficar fixas quando não apertamos nenhuma tecla, evitando que elas "pisquem". Ela só será apagada se houver movimento.

Teste estes programas:

```
1 LET X=10
2 LET Y=0
3 FOR A=0 TO 1000
4 LET X=X+(1 AND INKEY$="6")-
(1 AND INKEY$="7")
5 PRINT AT X,Y;"▒"
6 IF INKEY$="" THEN GOTO 6
7 CLS
8 NEXT A
```

Agora retire a linha 6.

Ela evita que o desenho seja apagado quando não há movimento.

Estes artifícios podem ser usados apenas em loops simples.

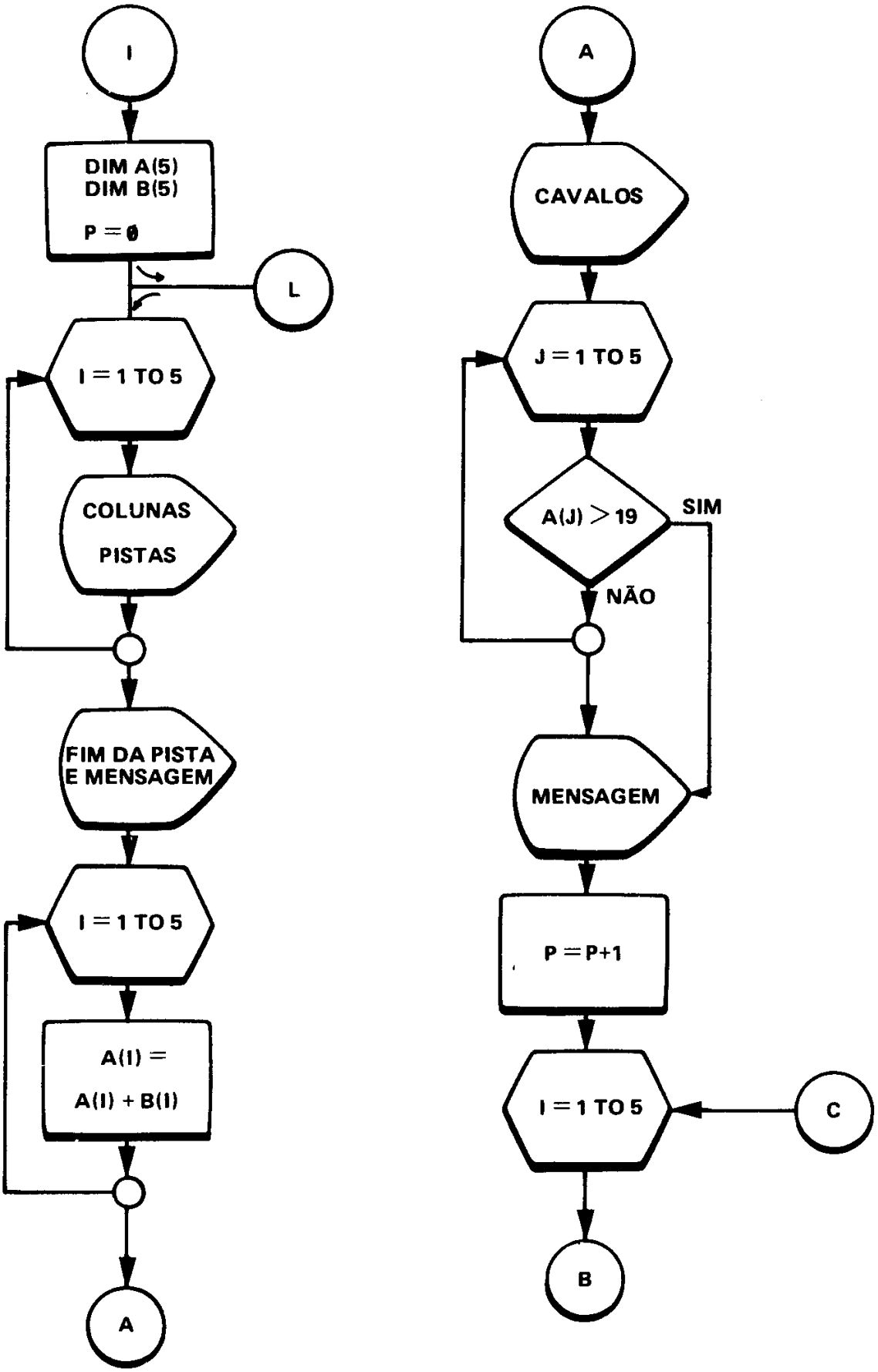
Para animação de movimento de bonecos, podemos utilizar alguns macetes como alternar caracteres:

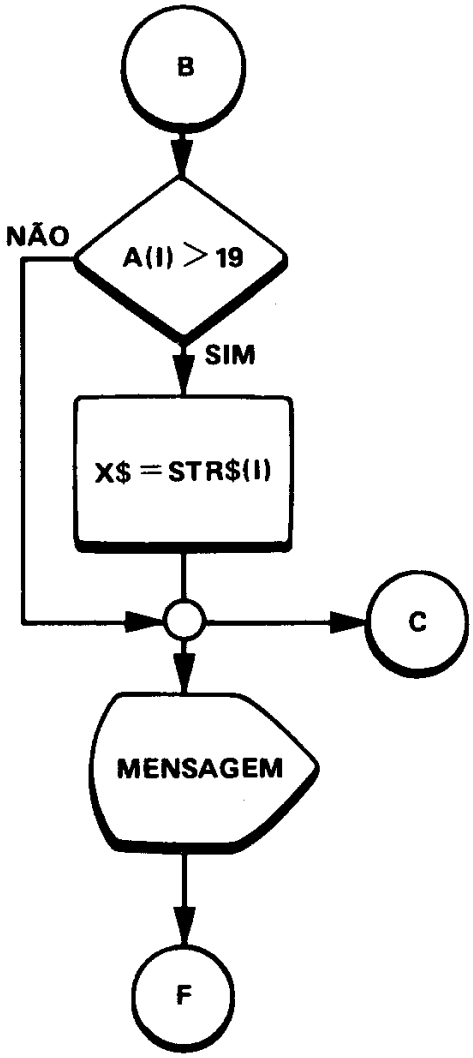
```
1 PRINT AT 10,10;"▒▒▒▒▒"
2 PRINT AT 10,10;"▒   ▒"
3 GOTO 1
```

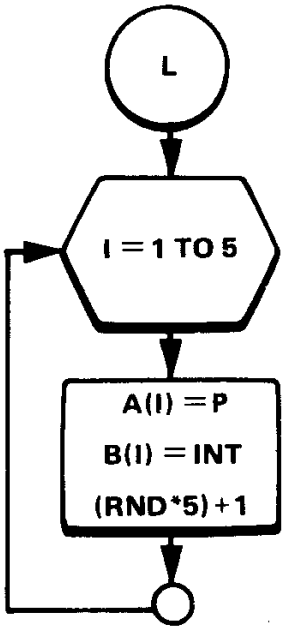
```
1 FOR A=0 TO 100
2 PRINT AT 10,10;"<▒>"
3 PAUSE 10
4 PRINT AT 10,10;">▒<"
5 PAUSE 10
6 NEXT A
```

JOCKEY

Neste jogo, apostamos num cavalo. Aleatoriamente, os cavalos correrão pela pista até cruzarem a reta final, quando será impressa uma mensagem.


I
DIM A(5)
DIM B(5)
P = 0
L
I = 1 TO 5
COLUNAS
PISTAS
FIM DA PISTA
E MENSAGEM
I = 1 TO 5
A(I) =
A(I) + B(I)
A
A
CAVALOS
J = 1 TO 5
A(J) > 19
SIM
NÃO
MENSAGEM
P = P+1
I = 1 TO 5
C
B


B
NÃO
A(I) > 19
SIM
X$ = STR$(I)
C
MENSAGEM
F


L
I = 1 TO 5
A(I) = P
B(I) = INT
(RND*5) + 1

JOCKEY

```
0 REM C-MICROMEGA 1983
2 DIM A(5)
3 DIM B(5)
5 SLOW
7 RAND
8 POKE 16418,0
10 LET P=PI-PI
30 FOR I=1 TO 5
32 LET A(I)=P
35 LET B(I)=INT (RND*3)+1
40 NEXT I
70 FOR L=2 TO 19
80 PRINT AT L,0;"▒  ▒  ▒  ▒  ▒
▒ ▒"
90 NEXT L
110 PRINT AT 0,0;"  1  2  3  4
5";AT 1,0;"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";A
T 21,0;"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";AT 20
,0;"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
120 PRINT AT 7,20;"FACAM SUAS";
AT 8,21;"APOSTAS"
130 PAUSE 300
140 PRINT AT 10,22;"ENCERRADAS"
;AT 11,20;"AS APOSTAS"
150 PAUSE 50
170 PRINT AT 13,20;"▒▒▒▒▒▒▒▒▒▒"
;AT 1,0;"▒▒ ▒▒ ▒▒ ▒▒ ▒▒ ▒▒"
200 FOR I=1 TO 5
205 LET A(I)=A(I)+B(I)
207 NEXT I
230 PRINT AT A(1),2;"1";AT A(2)
,5;"2";AT A(3),8;"3";AT A(4),11;
"4";AT A(5),14;"5"
240 FOR J=1 TO 5
242 IF A(J)>19 THEN GOTO 300
244 NEXT J
250 GOTO VAL "200"
300 PRINT AT 13,18;"▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒";AT 14,18;"▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
303 PAUSE 250
305 CLS
306 LET P=P+1
308 PRINT "ESTAMOS REVELANDO","
A FOTO DO PAREO NO. ";P
310 FOR I=1 TO 5
320 IF A(I)>19 THEN LET X$=STR$
I
330 NEXT I
500 PRINT ,,,,"O CAVALO ";X$;"
FOI O VENCEDOR,","PAGANDO ";(STR
$ (RND*5+1))( TO 3);" POR UM"
610 PAUSE 300
620 CLS
630 GOTO VAL "15"
```

TIROS:

O efeito do tiro de uma nave deve ser rápido e bonito e para isso temos várias escolhas:

a) Tiros feitos com pontos:

Fazemos o movimento de um ponto (.), desde o canhão até seu alcance máximo ou até atingir o alvo.
Tente:

```
10 PRINT AT 10,10;">"
20 FOR I=11 TO 25
30 PRINT AT 10,I;".";AT 10,I;"
"
40 NEXT I
```

Para que o tiro pare ao atingir o alvo, colocamos a coluna onde o alvo se encontra numa memória x e temos então duas opções:

```
20 FOR I=11 TO X
```

ou então:

```
35 IF I=X THEN STOP
```

b) Tiros feitos com PLOT:

Esse tiro já foi usado por nós no jogo "TIRO AO ALVO".

Usaremos agora o artifício do UNPLOT.
Tente:

```
10 FOR I=1 TO 62
20 PLOT I,20
30 UNPLOT I,20
40 NEXT I
```

O tiro é bonito, mas sua velocidade é muito baixa. Coloque agora:

```
10 FOR I=1 TO 62 STEP 5
```

Note que a velocidade aumenta sem que prejudiquemos o efeito.

d) Tiros em *strings*

Consistem num método mais complexo e muito bonito. Colocamos os tiros dentro de uma *string* e imprimimos um por um seus caracteres.

Tente:

```
10 LET A$=".....
20 LET Y=10
30 PRINT AT 10,Y;">"
40 FOR A=1 TO 5
50 PRINT AT 10,Y+A;A$(A)
60 NEXT A
```

```
10 LET A$="------"
20 PRINT AT 10,10;">"
30 FOR I=1 TO 5
40 PRINT AT 10,10+I;A$(I)
50 NEXT I
60 PRINT AT 10,11;"      "
```

EXPLOSÕES

Muito comum nos jogos, é que, quando atingimos um alvo ou somos atingidos, a nossa nave ou o alvo devem explodir.

Desenhar uma explosão é muito fácil. Podemos, por exemplo, cobrir o alvo ou a nave de caracteres cinza.

Para um bonito efeito na explosão, é muito comum que escrevamos "BUM" ao lado da nave, e fazemos este "BUM" piscar.

Tente:

```
10 FOR I=1 TO 20
20 PRINT AT 10,10;"BUM"
30 PRINT AT 10,10;"   "
40 NEXT I
```

Outro efeito muito bonito é a projeção de estilhaços a partir do alvo ou da nave que explodiu.

Teste o programa:

```
  10 FOR I=0 TO 5
  20 PRINT AT 10+I,10;".";AT 10+
I,10-I;".";AT 10+I,10+I;"."
  30 NEXT I
```

LIMPANDO A TELA

Após uma explosão ou uma outra coisa qualquer, é muito

comum termos que limpar apenas um pedaço da tela, sem afetarmos os outros desenhos.

O artifício usado (já apareceu no "Salvamento") é de cobrirmos esse pedaço da tela com caracteres brancos.

Teste o programa:

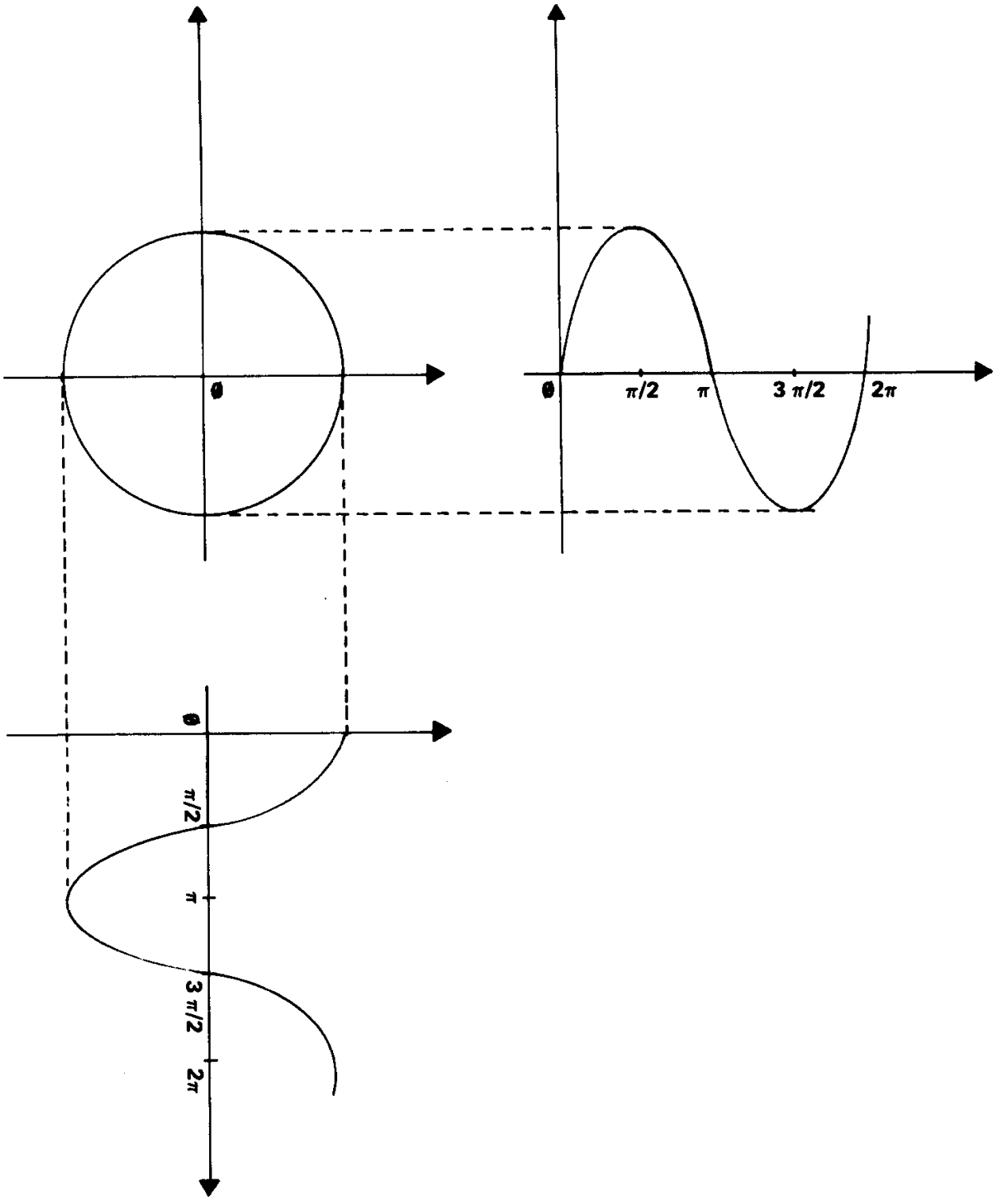
```
10 FOR L=0 TO 21
20 FOR C=0 TO 31
30 PRINT AT L,C;"±"
40 NEXT C
50 NEXT L
60 FOR A=5 TO 15
70 PRINT AT A,10;"
80 NEXT A
```

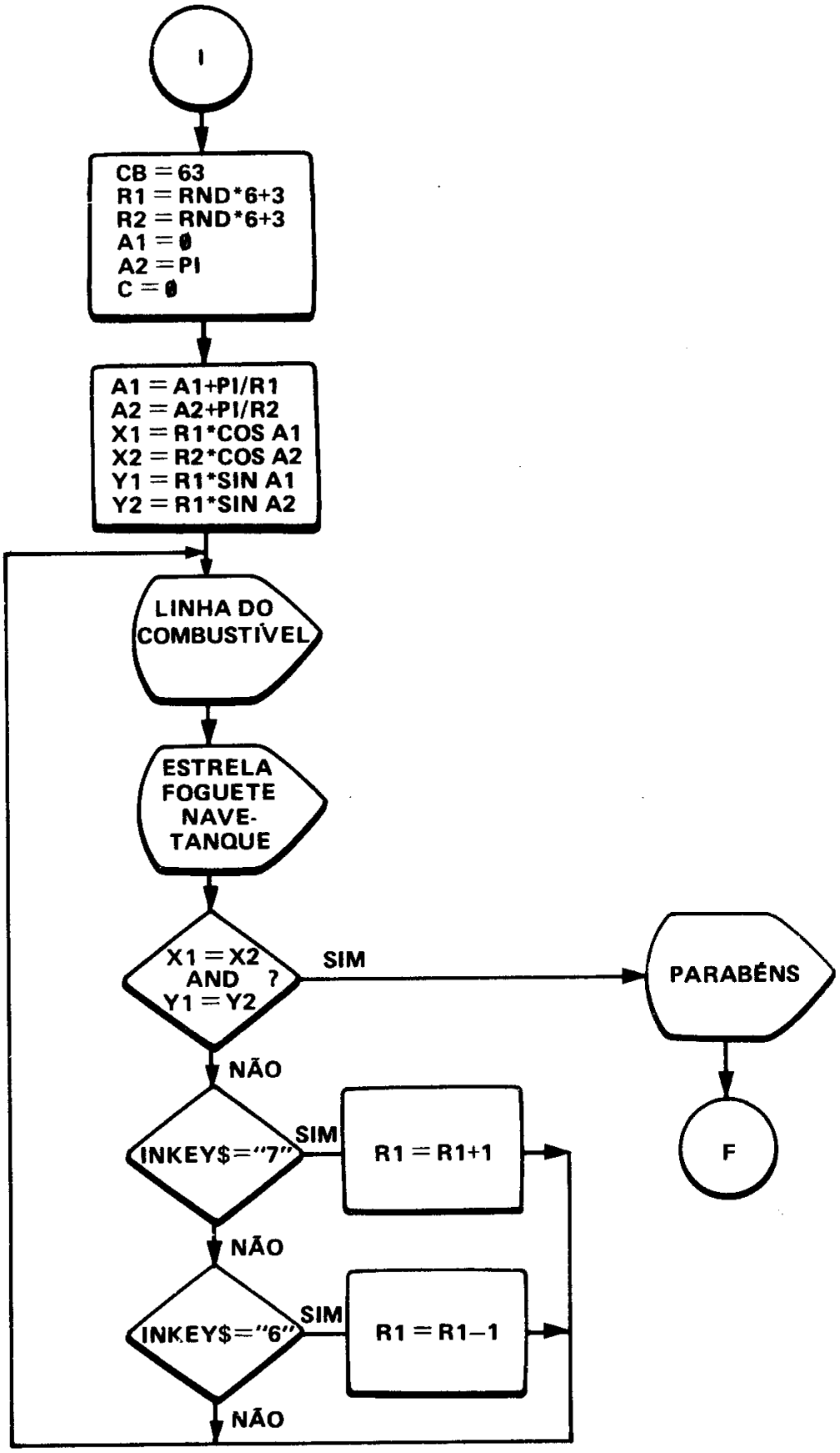
Perceba que apagamos apenas um quadrado central na tela.

ÓRBITA

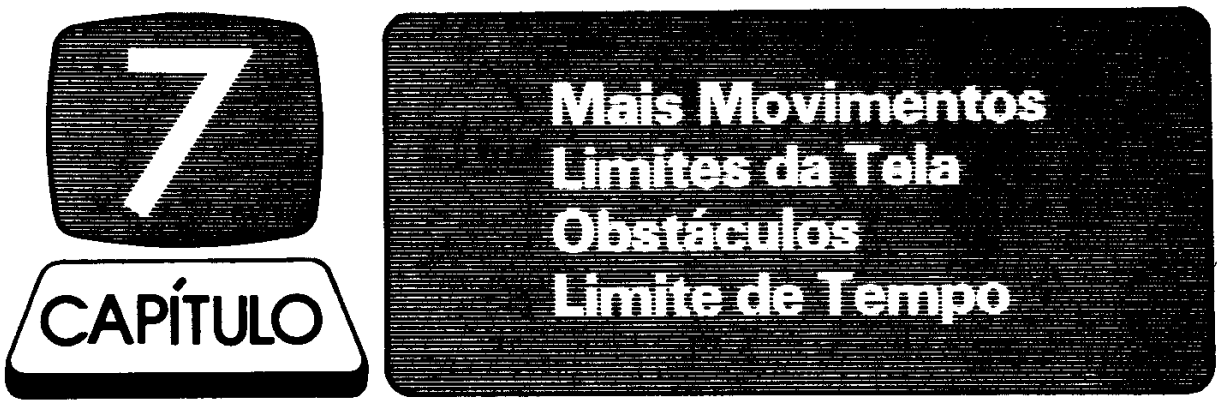
Neste jogo aparece pela primeira vez o movimento circular.

Este movimento é conseguido com a composição de um SENO no eixo X e um COSSENO no eixo Y. Observe o gráfico explicativo:


0
0
π/2
π
3 π/2
2π
0
π/2
π
3 π/2
2π


I
CB = 63
R1 = RND*6+3
R2 = RND*6+3
A1 = 0
A2 = PI
C = 0
A1 = A1+PI/R1
A2 = A2+PI/R2
X1 = R1*COS A1
X2 = R2*COS A2
Y1 = R1*SIN A1
Y2 = R1*SIN A2
LINHA DO COMBUSTÍVEL
ESTRELA FOGUETE NAVE-TANQUE
X1 = X2 AND Y1 = Y2 ?
SIM
PARABÉNS
F
NÃO
INKEY$="7"
SIM
R1 = R1+1
NÃO
INKEY$="6"
SIM
R1 = R1–1
NÃO

```
  1 REM "ORBITA"
  2 RAND
  5 LET CB=63
 10 LET A1=INT (RND*6)+3
 15 LET A2=INT (RND*6)+3
 20 LET A1=0
 25 LET A2=PI
 30 LET C=0
 32 GOSUB 70
 35 LET A1=A1+PI/A1
 40 LET A2=A2+PI/A2
 45 LET X1=INT (A1*COS A1+16.5)
 50 LET X2=INT (A2*COS A2+16.5)
 55 LET Y1=INT (A1*SIN A1+10.5)
 60 LET Y2=INT (A2*SIN A2+10.5)
 65 GOTO 100
 70 PRINT "O R B I T A"
 75 PRINT AT 20,0;"COMBUSTIVEL:
"
 80 FOR I=0 TO CB
 85 PLOT I,0
 90 NEXT I
 95 PRINT AT 10,16;"*"
 97 RETURN
100 PRINT AT Y1,X1-1;"▗▀▖"
105 IF X1=X2 AND Y1=Y2 THEN GOT
O 900
110 PRINT AT Y2,X2;"▣"
115 LET A1=A1+(1 AND INKEY$="7"
)-(1 AND INKEY$="6")
120 IF A1>8 THEN LET A1=8
125 IF A1=1 THEN LET A1=2
130 UNPLOT 63-C,0
140 LET C=C+1
150 IF C=CB THEN GOTO 500
155 PRINT AT Y1,X1-1;"   "
157 PRINT AT Y2,X2," "
160 GOTO 35
500 PRINT AT 10,6;"SEU COMBUSTI
VEL ACABOU";AT 11,7;"FIM DO JOGO
"
510 STOP
900 PRINT AT Y2,X2;"▣"
905 PRINT AT 10,6;"PARABENS..."
;AT 11,6;"RESTAVAM ";CB-C;" L DE
COMB."
910 GOTO 75
999 REM ** PROGRAMA ORIGINAL DE
LUZO DANTAS **
```



MOVIMENTOS

Movimentar a nave para as quatro direções é muito simples: basta incrementar ou decrementar as coordenadas X e/ou Y em função de uma tecla apertada (INKEY$).

Devemos porém, tomar certos cuidados tais como apagar a nave anterior antes de alterarmos as coordenadas X e/ou Y da "nave". Outro cuidado é o uso dos artifícios para apagar a nave, que deverão ser bem pensados já que se trata agora de um movimento nas quatro direções.

LIMITE DE TELA

Um dos problemas que enfrentamos durante o movimento de uma nave, é a possibilidade do rebatimento da coordenada ou o famoso erro "5".

Isto ocorre porque o movimento ultrapassa os limites da tela.

Uma das soluções encontradas é a passagem de cima para baixo, de baixo para cima, da esquerda para a direita ou da direita para a esquerda diretamente pelas bordas da tela.

Isto é conseguido transformando as coordenadas quando elas ultrapassarem os limites.

Tente:

```
 10 LET X=10
 20 LET Y=10
 30 PRINT AT X,Y;"*"
 40 PRINT AT X,Y;" "
 50 LET X=X+(1 AND INKEY$="6")-
(1 AND INKEY$="7")
 60 LET Y=Y+(1 AND INKEY$="8")-
(1 AND INKEY$="5")
 70 IF X>21 THEN LET X=0
 80 IF X<0 THEN LET X=21
 90 IF Y>31 THEN LET Y=0
100 IF Y<0 THEN LET Y=31
110 GOTO 30
```

Outros artifícios possíveis são:

Evitar que a nave passe os limites

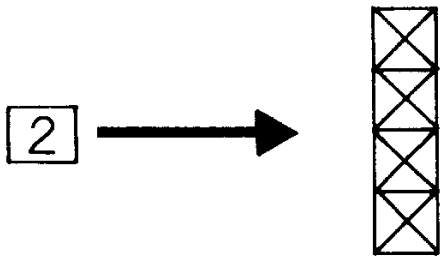
```
IF X>21 THEN LET X=21
```

Ou explodir a nave caso o limite seja ultrapassado

```
IF X>21 THEN GOTO (EXPLOSÃO)
```

OBSTÁCULOS

Colocar um obstáculo é a mesma coisa que limitar uma área dentro do campo de ação da nave.



Usamos, para tal, um artifício parecido com o do limite de tela.

Considere XO e YO as coordenadas do obstáculo e X e Y as da nave. A proibição de movimento será na linha:

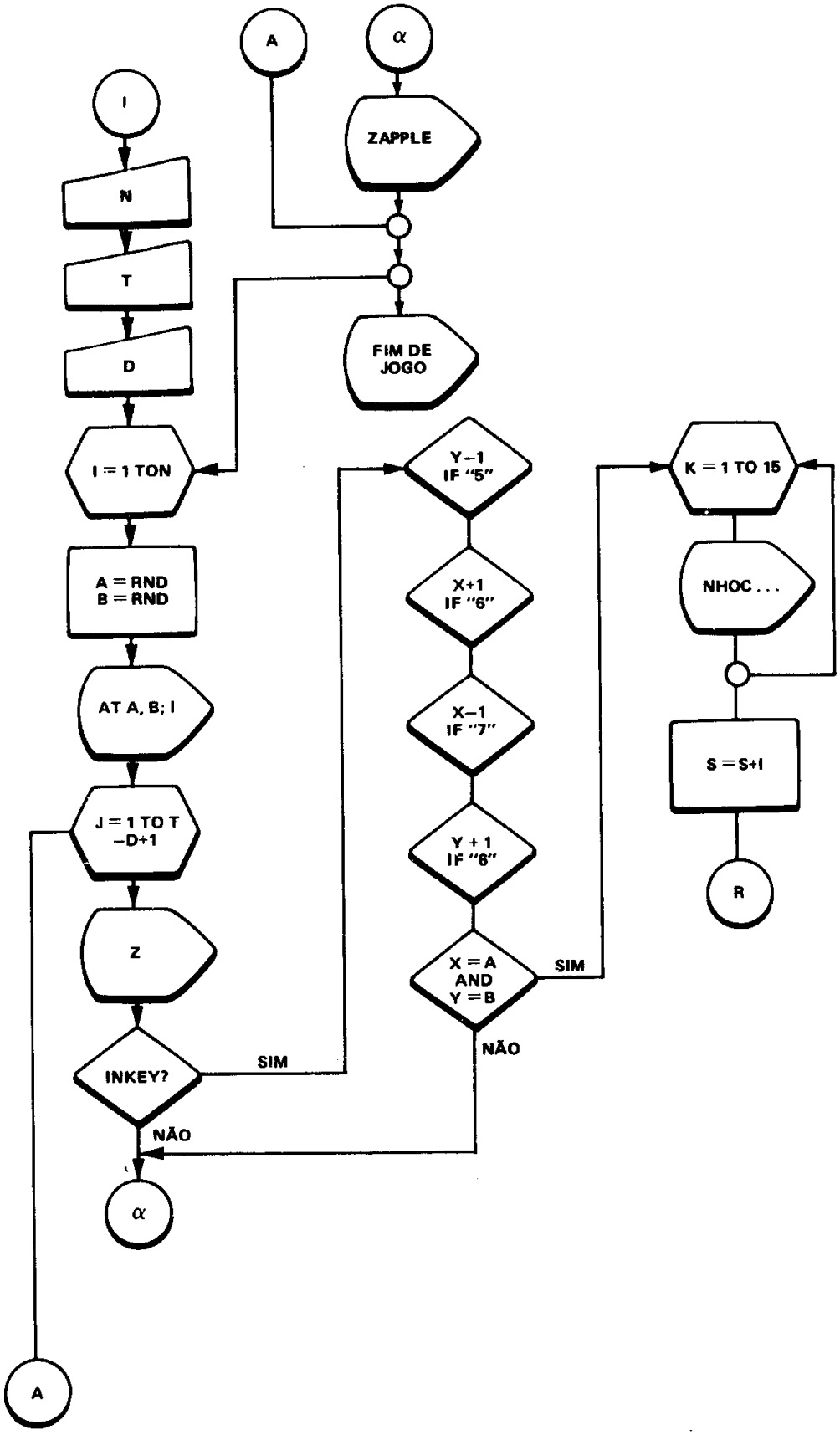
```
IF X=XO AND Y=YO THEN GOTO ...
```

LIMITE DO TEMPO

Muitas vezes limitamos, ou melhor, determinamos um certo tempo para que algo seja realizado. Esse tempo não é o mesmo do *loop* total do jogo.

COME NÚMERO

Neste jogo seu objetivo é "comer" números que vão aparecer na tela. Para cada número você tem um tempo certo para comê-los.


A
α
I
ZAPPLE
N
T
D
FIM DE JOGO
I = 1 TON
A = RND
B = RND
AT A, B; I
J = 1 TO T −D+1
Z
INKEY?
SIM
NÃO
α
A
Y−1 IF "5"
X+1 IF "6"
X−1 IF "7"
Y + 1 IF "6"
X = A AND Y = B
SIM
NÃO
K = 1 TO 15
NHOC . . .
S = S+I
R

```
    2  LET X=10
    3  LET Y=10
    5  RAND
   10  PRINT "QUANTOS NUMEROS ?"
   15  INPUT N
   20  PRINT "TEMPO EM UTUS ?"
   25  INPUT T
   30  PRINT "DIFICULDADE ? (1/2/3
)"
   35  INPUT D
   40  CLS
   45  LET S=0
   50  PRINT "PLACAR:";S
   55  FOR I=1 TO N
   60  LET A=INT (RND*21)+1
   65  LET B=INT (RND*32)+1
   70  PRINT AT A,B;I
   75  FOR J=1 TO T-D*I
   80  PRINT AT X,Y;"■"
   85  PRINT AT X,Y;" "
   90  IF INKEY$="5" THEN LET Y=Y-
1
   95  IF INKEY$="8" THEN LET Y=Y+
1
  100  IF INKEY$="7" THEN LET X=X-
1
  105  IF INKEY$="6" THEN LET X=X+
1
  110  IF X<1 THEN LET X=21
  120  IF X>21 THEN LET X=1
  125  IF Y<0 THEN LET Y=31
  130  IF Y>31 THEN LET Y=0
  140  IF X=A AND Y=B THEN GOSUB 5
00
  155  NEXT J
  160  NEXT I
  170  CLS
  175  PRINT AT 10,10;"FIM DE JOGO
"
  180  IF S>50 THEN PRINT AT 13,12
;"MUITO BOM"
  190  STOP
  500  LET S=S+I
  501  PRINT AT A,B;"■"
  505  PRINT AT 0,7;S
  510  FOR K=1 TO 15
  515  PRINT AT 0,24;"NHOC..."
  520  PRINT AT 0,24;"■■■■■■■■■■"
  525  NEXT K
  530  PRINT AT 0,24;"          "
  535  LET J=T-D*I+2
  536  PRINT AT A,B;" "
  540  RETURN
```

Cuidados na Preparação dos Jogos
Dicas
Joystick
Desenhos Úteis
Conferindo letras

CUIDADO NA PREPARAÇÃO DOS JOGOS

— Limite sempre a tela, isso evitará o erro "5".

— Cuidado ao decrementar tempo (veja capítulo 4)

— Cuidado quando "criar" a função para o nível de dificuldade. Não crie coisas absurdas.

— Coloque sempre instruções.

— Não copie jogos já prontos. Invente os seus próprios.

Dicas:

— Use sempre INKEY$ 5, 6, 7, 8 e 0, isto possibilita o uso de JOYSTICK.

— Economize tempo

— Não faça operações complexas em muitas linhas. Reduza o número de linhas ao máximo.

— Economize memória.

— Substitua (quando necessário)

```
GOTO 100


GOTO VAL "100"

LET A=0

LET A=PI-PI

LET A=1

LET A=PI/PI


IF INKEY$="6" THEN LET A=A+1

IF INKEY$="7" THEN LET A=A-1


LET A=A+1-(1 AND INKEY$="6")-(1 A
ND INKEY$="7")
```

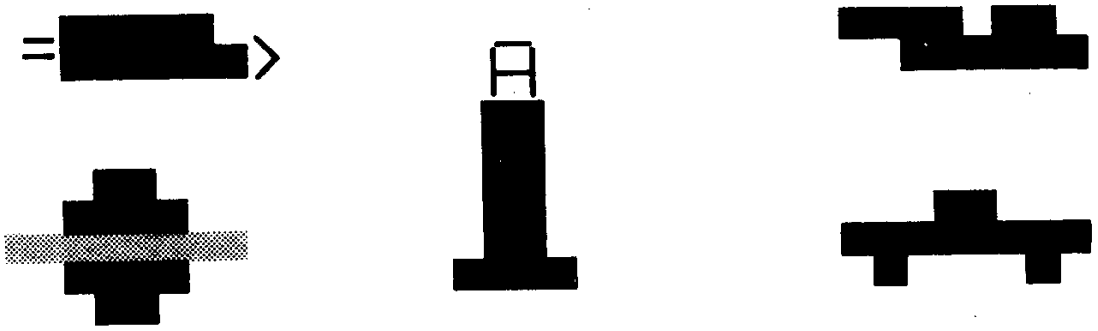
Isso economizará memória.

— Para usar 23 linhas de tela, use:

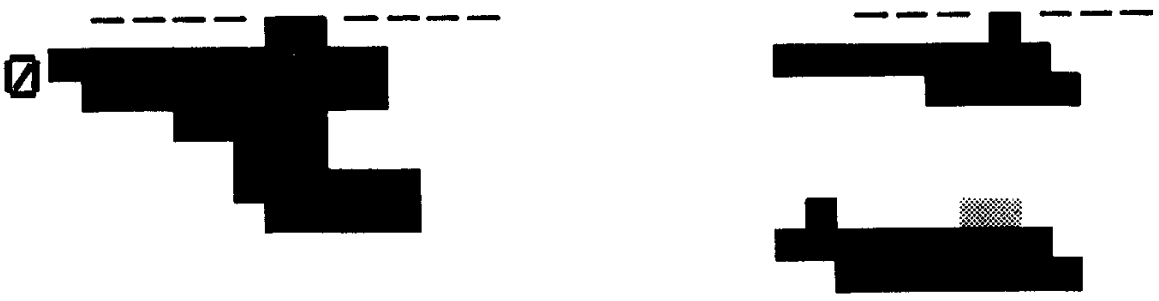
```
POKE 16418,0
```

DESENHOS ÚTEIS NOS JOGOS:

1) Naves Espaciais



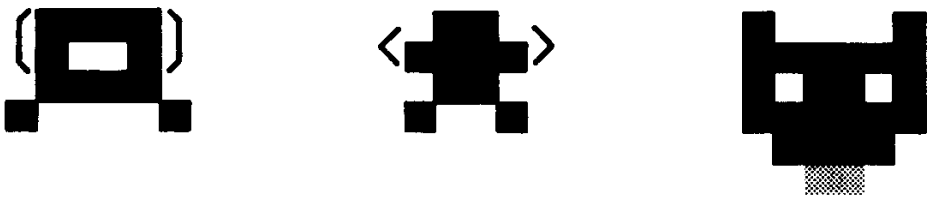
2) Aviões e Helicópteros



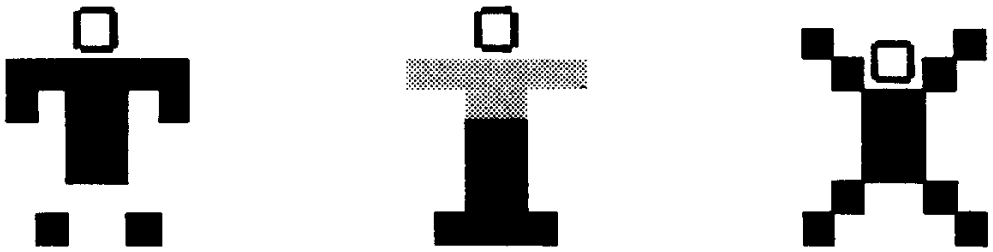
3) Barcos

4) Carros

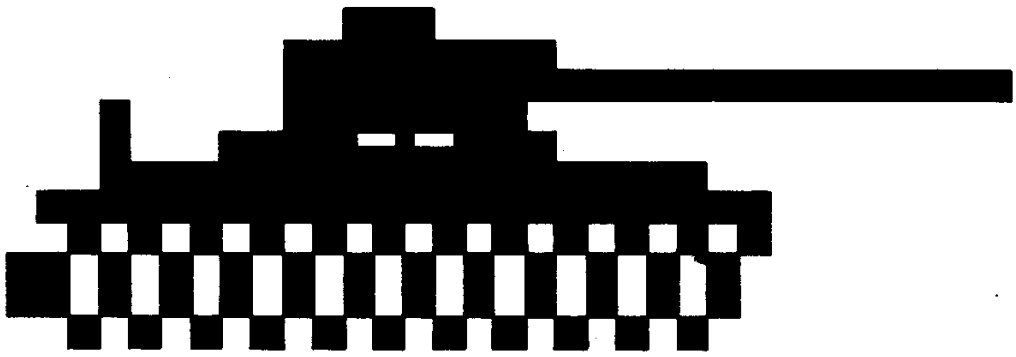
5) Invasores



6) Pessoas ou Robos



7) Tanque



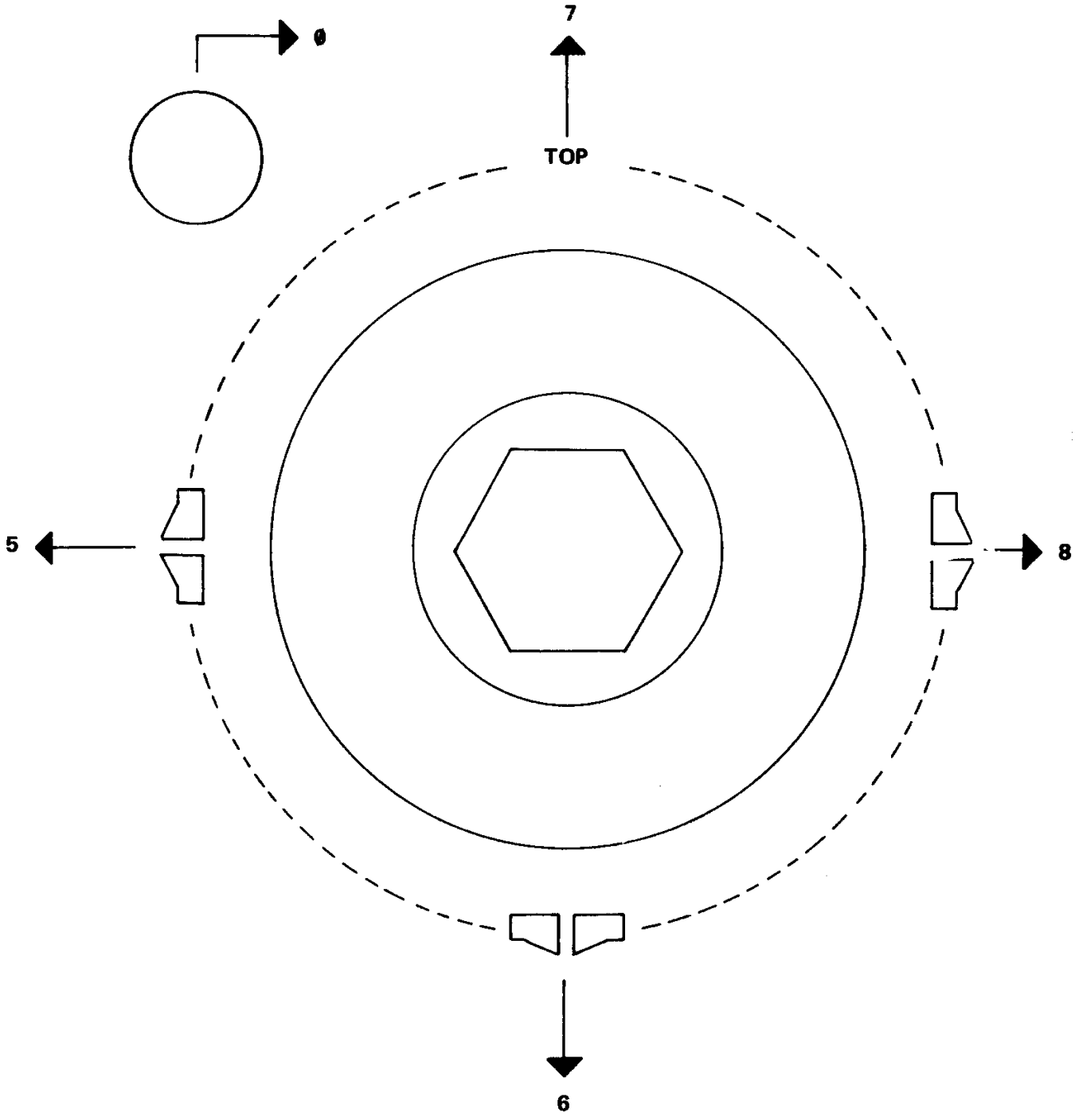
JOYSTICK

Para que num jogo possamos usar o Joystick, devemos programá-lo para que os controles sejam feitos pelas teclas 5, 6, 7, 8 e 0.

No programa, usamos a função INKEY$.

O Joystick funciona exatamente como se tivéssemos apertado as teclas.

Veja o desenho:



CONFERINDO LETRAS

Um jogo bastante conhecido é o jogo da forca. O programa que desenvolvemos escolhe aleatoriamente uma palavra e pede letras ao jogador. A cada letra digitada, ele confere se ela existe na palavra ou se já foi digitada.

Neste jogo, teremos que fazer algo diferente do que fizemos até agora: devemos conferir se uma letra digitada faz parte da palavra escolhida. O problema é: como fazer isso se a palavra é uma *string* inteira? O truque é desmembrar a *string*. Temos então:

```
A=VAL P$              (P$ - INCOGNITA)

INPUT L$              (L$ - LETRA)

FOR I=1 TO A

IF L$=A$(I) THEN GOSUB  ALFA
(ALFA - SUBROTINA DE ACERTO)
NEXT I
```

Além disso temos que conferir a linha digitada. Isso é feito por:

```
VA=1     (VA - VARIAVEL CONTADORA)
                        INICIO
DIM B$(30)   (A$ - ARQUIVO DE DI-
                          GITACAO)
LET B$(VA)=L$   (ARMAZENA LETRAS)
LET VA=VA+1    (INCREMENTA CONTA-
                            DORA)

FOR I=1 TO VA
IF L$=B$(I) THEN GOTO BETA
NEXT I

   CONFERE ANTES
   DE ARMAZENAR
(BETA=LETRA IGUAL)
```

Esta é a chave do jogo da forca: na linha DIM A$ (30), o número exato para(n) é 23, pois nosso alfabeto tem 23 letras e não poderemos guardá-las duas vezes na matriz A$.

```
 1 REM PROGRAMA EXEMPLO
 2 LET VA=1
 5 DIM A$(5,10)
10 GOSUB 100
15 LET A=INT (RND*5)+1
20 LET P$=A$(A)
25 INPUT L$
30 GOSUB 150
35 FOR I=1 TO 10
40 IF L$=P$(I) THEN GOSUB 250
```

```
 45 NEXT I
 50 GOTO 25
100 LET A$(1)="ENGENHEIRO"
110 LET A$(2)="TECNOLOGIA"
120 LET A$(3)="ASTRONOMIA"
130 LET A$(4)="OLIGARQUIA"
140 LET A$(5)="ARITHETICA"
150 RETURN
160 DIM B$(24)
170 FOR J=1 TO VA
180 IF L$=B$(J) THEN GOTO 300
190 NEXT J
200 LET B$(VA)=L$
210 LET VA=VA+1
220 RETURN
250 PRINT AT 10,5+I;L$
260 RETURN
300 PRINT AT 20,0;"LETRA JA DIG
ITADA"
310 PAUSE 100
315 PRINT AT 20,0;"
          "
320 GOTO 25
```

Agora apresentamos um jogo da forca completo, onde você pode jogar contra um colega e o computador se encarregará de montar o "boneco" na forca.

```
145 CLS
150 PRINT "JOGO PARA 2 PESSOAS"
,,
152 PRINT "PALAVRA COM ATE 9 LE
RAS",,,
155 PRINT "QUEM COMECA ?"
156 INPUT N$
157 PRINT N$
160 PRINT "E O NOME DE SEU OPON
ENTE ?"
161 INPUT M$
163 PRINT M$
165 PAUSE 90
166 CLS
167 LET N=0
168 LET M=0
169 LET B$=""
170 FOR W=1 TO 2
171 CLS
```

```
 172 IF W=2 THEN PRINT M$;" DIGI
TE A PALAVRA"
 173 IF W=1 THEN PRINT N$;" DIGI
TE A PALAVRA"
 175 INPUT Q$
 176 IF LEN Q$>9 THEN GOTO 175
 177 LET Z=LEN Q$
 178 PRINT Q$,,,,'
 179 PRINT "ESTA CORRETA ? (S/N)
"
 185 INPUT R$
 187 IF R$="N" THEN GOTO 171
 190 CLS
 191 PRINT AT 0,0;"LETRAS  DIGIT
ADAS";AT 1,0;"--------------------"
;AT 9,4;"?????????"( TO Z)
 192 LET S=0
 193 LET C=0
 194 LET B$=""
 200 FOR G=1 TO 17
 210 PRINT AT 15,2;"DIGITE A LET
RA"
 220 INPUT P$
 221 PRINT AT 15,2;"
   "
 223 IF P$<"A" OR P$>"Z" THEN GO
TO 210
 224 LET P$=P$(1)
 225 IF G=1 THEN GOTO 242
 227 LET K=0
 230 FOR H=1 TO G-1
 235 IF P$<>B$(H) THEN LET K=K+1
 240 NEXT H
 241 IF K<G-1 THEN GOTO 700
 242 LET B$=B$+P$
 245 PRINT AT 1,G-1;P$
 246 LET J=0
 250 FOR D=1 TO Z
 255 IF P$<>Q$(D) THEN LET J=J+1
 260 NEXT D
 265 IF J=Z THEN GOTO 900
 270 FOR T=1 TO Z
 275 IF P$=Q$(T) THEN PRINT AT 9
 T+3;P$
 280 IF P$=Q$(T) THEN LET S=S+1
 285 NEXT T
 290 IF S=Z THEN GOTO 500
 295 NEXT G
 500 FOR O=1 TO 30
 510 PRINT AT 12,4;"CONSEGUIU"
 515 PRINT AT 12,4;"▒▒▒▒▒▒▒▒▒▒▒"
 520 NEXT O
 530 IF W=1 THEN LET M=M+1
 535 IF W=2 THEN LET N=N+1
 550 CLS
 560 PRINT M$;":  ";M;" PONTOS"
 561 PRINT N$;":  ";N;" PONTOS"
 575 PAUSE 100
 580 NEXT W
```

```
 700 PRINT AT 16,3;"A LETRA ";CH
R$ (CODE P$+128);AT 17,0;"JA FOI
 DIGITADA"
 710 PAUSE 120
 720 PRINT AT 16,3;"              ";
AT 17,0;"                     "
 730 GOTO 210
 900 LET C=C+1
 905 GOTO 1000*C
1000 FOR V=21 TO 1 STEP -1
1010 PRINT AT V,18;"█"
1020 NEXT V
1030 PRINT AT 0,18;"█████████"
1040 GOTO 295
2000 PRINT AT 2,23;" █████ "
2010 PRINT AT 3,23;"█ █ █ █"
2020 PRINT AT 4,23;" █ ( ) █"
2030 PRINT AT 5,23;" █ ███ █"
2040 PRINT AT 6,23;" █   █"
2050 PRINT AT 7,23;" ███"
2060 GOTO 295
3000 PRINT AT 8,23;"███████"
3010 PRINT AT 9,23;"███████"
3020 PRINT AT 10,23;" ██████ "
3030 PRINT AT 11,23;" ██████ "
3040 PRINT AT 12,23;" ██████ "
3050 PRINT AT 13,23;" ██████ "
3055 PRINT AT 14,23;" ██████ "
3060 GOTO 295
4000 PRINT AT 8,22;"█"
4010 FOR V=9 TO 12
4020 PRINT AT V,22;"█"
4030 NEXT V
4040 PRINT AT 13,21;" █"
4050 PRINT AT 14,21;" █"
4060 GOTO 295
5000 PRINT AT 8,29;"█"
5010 FOR V=9 TO 12
5020 PRINT AT V,29;"█"
5030 NEXT V
5040 PRINT AT 13,29;"█ "
5050 PRINT AT 14,29;"█"
5060 GOTO 295
6000 FOR V=15 TO 19
6010 PRINT AT V,23;" ██ "
6020 NEXT V
6030 PRINT AT 16,24;"█"
6040 PRINT AT 17,24;"█"
6050 GOTO 295
7000 FOR V=15 TO 19
7010 PRINT AT V,26;" ██ "
7020 NEXT V
7030 PRINT AT 16,27;"█"
7040 PRINT AT 17,27;"█"
7050 GOTO 295
8000 PRINT AT 20,23;" █   █ "
8001 PRINT AT 3,1;"████████████
██";AT 4,1;"████████████████"
8010 GOTO 295
```

```
9000 PRINT AT 1,25;" ■"
9010 PRINT AT 5,25;" ■ "
9011 PRINT AT 3,1;"
   ";AT 4,1;"                    "
9020 PAUSE 120
9030 PRINT AT 7,3;"A PALAVRA E :
";AT 6,13;"/"
9040 PRINT AT 9,4;Q$
9050 IF W=1 THEN LET N=N+1
9060 IF W=2 THEN LET M=M+1
9065 PAUSE 180
9070 GOTO 550
```

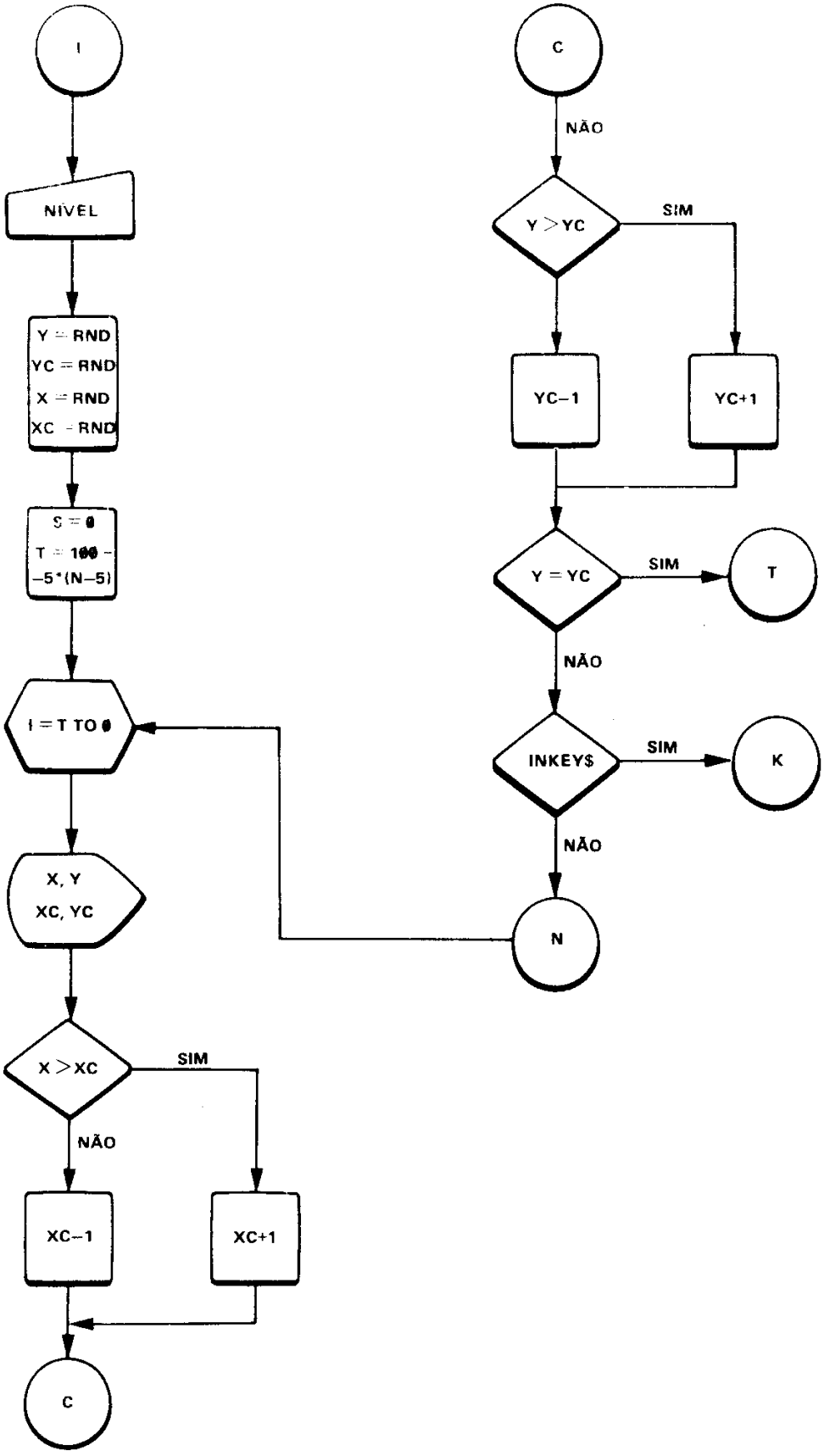
TANK

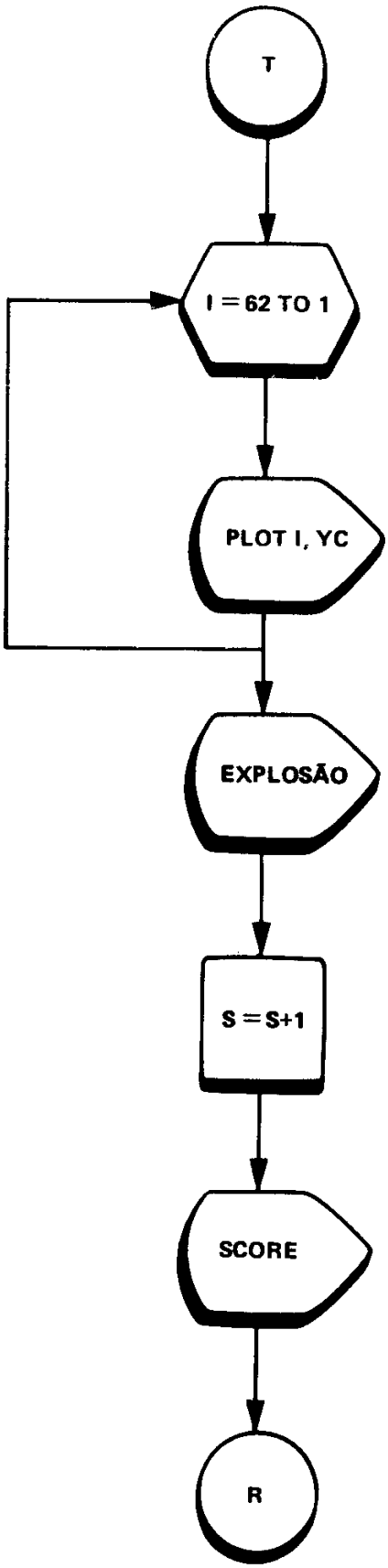
Neste jogo veremos uma série de procedimentos discutidos ao longo deste livro.Preste atenção tanto no diagrama de blocos quanto na listagem para analisar como procedemos na elaboração deste jogo.

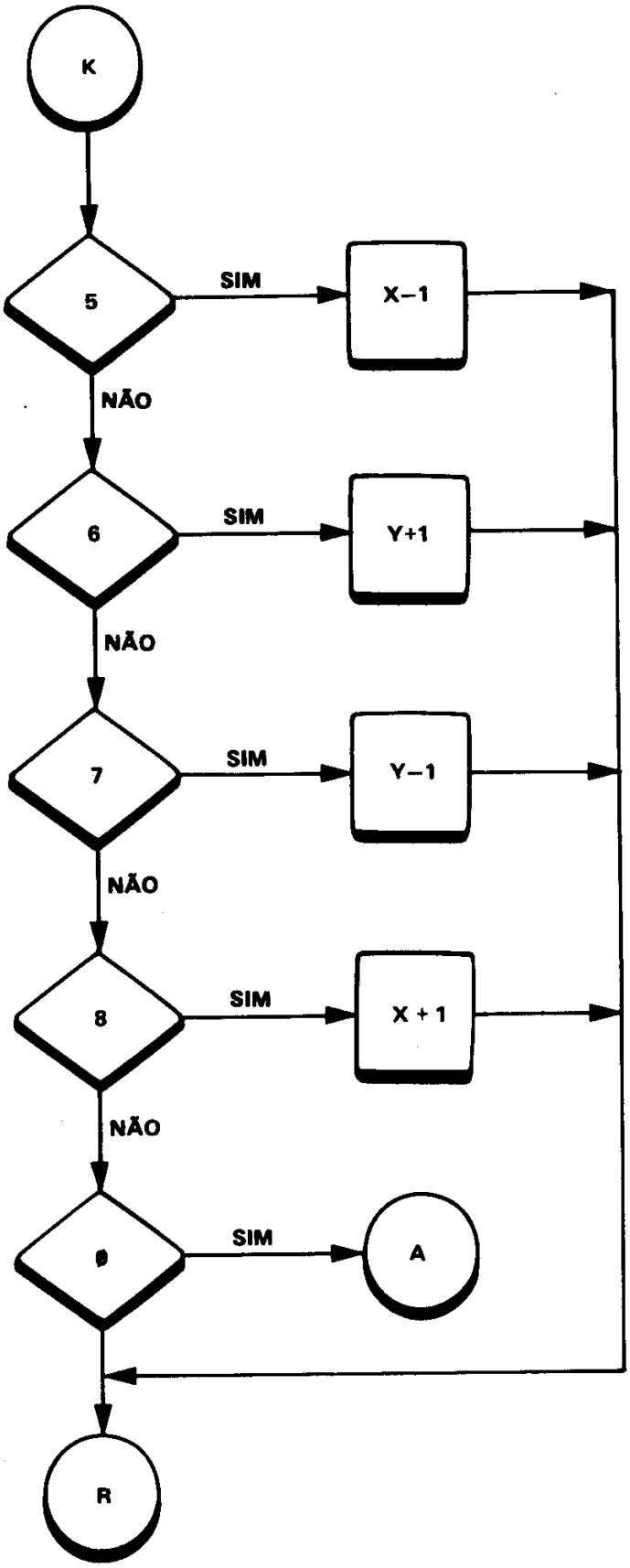
```
   0 REM C-MICROMEGA   1983
   0 REM FABIO RENDELUCCI
   1 REM "TANK"
   2 CLS
   3 RAND
   5 SLOW
  60 PRINT AT 5,6;"▓▓▓▓ ▓▓▓▓ ▓▓▓▓ ▓
 ▓";AT 9,9;"▓ ▓▓ ▓▓▓ ▓▓▓ ";AT 1
0,9;"▓  ▓▓▓ ▓▓▓▓ ▓▓▓";AT 11,9;"▓
▓▓ ▓▓▓ ▓▓▓ ";AT 12,9;"▓  ▓▓ ▓▓ ▓▓
 ▓ ▓▓"
 120 PAUSE 50
 130 CLS
 135 PRINT AT 2,6;"* INSTRUCOES
*";AT 5,2;"USE <7> PARA SUBIR";A
T 7,2;"USE <6> PARA DESCER";AT 9
,2;"PARA ATIRAR,PRESSIONE <0>";A
T 11,2;"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
▓▓▓▓";AT 18,2;"APERTE <1> PARA CO
MECAR"
 140 PAUSE 300
 145 POKE 16437,255
 150 IF NOT INKEY$="1" THEN GOTO
 1
 160 CLS
 170 POKE 16416,2
 175 PRINT AT 2,2;"NIVEL ?  (1/2
/3/4/5)"
 180 INPUT N
 185 CLS
 190 LET CR=0
```

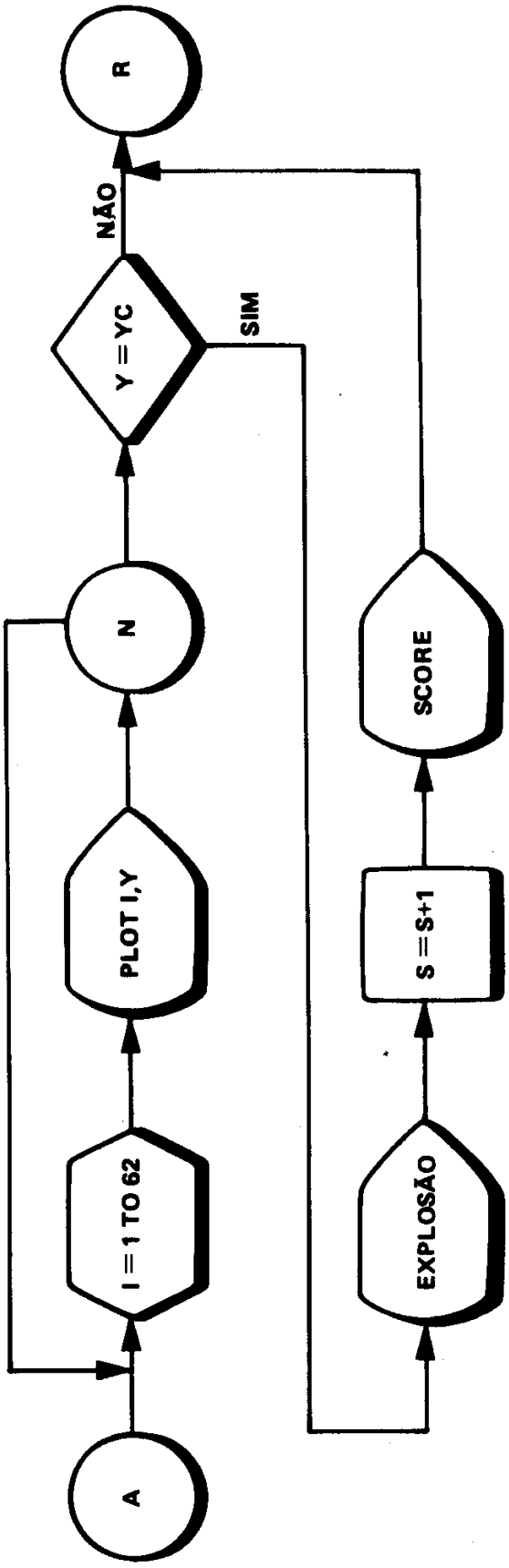
```
 195 LET CB=0
 200 LET T=300-INT (10*N)
 205 PRINT AT 1,11;"  TANK"
 210 POKE 16418,0
 215 PRINT AT 2,4;"NIVEL:";N;AT
2,18;"TEMPO:";T;AT 3,11;"PLACAR:
";AT 3,4;CA;AT 3,27;CB
 220 FOR A=0 TO 31
 225 PRINT AT 4,A;"▀";AT 22,A;"▄
"
 230 NEXT A
 240 REM ** JOGO **
 245 LET YT=INT (RND*8)+5
 250 LET YC=INT (RND*10)+5
 252 LET J=T
 253 FOR Q=0 TO J
 254 PRINT AT YT,0;" ";AT YT+1,0
;" ";AT YC,31;" ";AT YC+1,31;" "
 255 LET YT=YT+(1 AND INKEY$="6"
)-(1 AND INKEY$="7")
 265 IF YT<YC THEN LET YC=YC-1
 270 IF YT>YC THEN LET YC=YC+1
 280 PRINT AT YT,0;"▄";AT YT+1,0
;"▀";AT YC,31;"▄";AT YC+1,31;"▀"
 282 IF INKEY$="0" THEN GOSUB 10
00
 285 IF (YC=YT OR YC=YT+1) AND (
INT ((N*10)*RND)>5) THEN GOSUB 1
500
 290 LET T=T-1
 293 PRINT AT 2,24;"    "
 295 PRINT AT 2,24;T
 300 NEXT Q
 310 FOR A=0 TO 20
 315 NEXT A
 320 CLS
 325 PRINT "OUTRA VEZ ? (S/N)"
 330 INPUT H$
 335 IF H$="N" THEN NEW
 340 CLS
 345 RUN
1000 REM ** TIRO T **
1005 FOR A=1 TO 62 STEP 6
1010 PLOT A,(43*(YT-21))/-21
1015 UNPLOT A,(43*(YT-21))/-21
1020 NEXT A
1025 IF YT=YC OR YT=YC+1 THEN GO
SUB 2000
1030 RETURN
1500 REM ** TIRO C **
1505 FOR A=62 TO 1 STEP -6
1510 PLOT A,(43*(YC-21))/-21
1515 UNPLOT A,(43*(YC-21))/-21
1520 NEXT A
1530 PRINT AT YT,0;"▒▒▒";AT YT-1,
0;"▒";AT YT+1,0;"▒"
1532 FOR A=0 TO 30
1535 PRINT AT YT,4;"BUM"
1540 PRINT AT YT,4;"▒▒▒"
1545 NEXT A
```

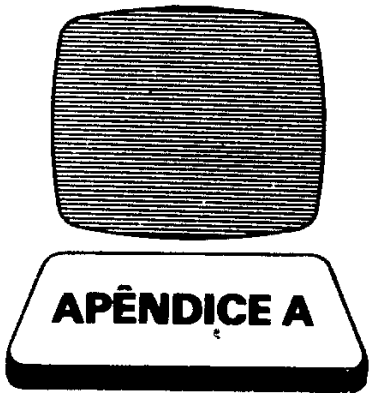
```
1550 LET CB=CB+10
1555 PRINT AT 3,27;CB
1557 GOSUB 2500
1560 RETURN
2000 REM ** EXPLOSAO C **
2005 PRINT AT YC,30;"▒▒▒";AT YC-1
,31;"▒";AT YC+1,31;"▒"
2010 FOR A=0 TO 30
2015 PRINT AT YC,25;"BUM"
2020 PRINT AT YC,25;"▒▒▒"
2025 NEXT A
2030 LET CA=CA+10
2035 PRINT AT 3,4;CA
2037 GOSUB 2600
2040 RETURN
2500 REM ** LIMPEZA **
2505 PRINT AT YT,0;"                ";
AT YT+1,0;"   ";AT YT-1,0;"   "
2510 PRINT AT YC+2,29;"   ";AT Y
C+1,29;"   ";AT YC-1,31;" ";AT Y
C,30;"  "
2511 LET YT=YT+3
2512 LET YC=YC-3
2515 RETURN
2600 REM ** LIMPEZA C **
2605 PRINT AT YC+2,29;"   ";AT Y
C+1,29;"   ";AT YC-1,29;"   ";AT
 YC,25;"       ";AT YT,0;" ";AT
YT-1,0;" ";AT YT+1,0;" "
2607 LET YT=YT-3
2608 LET YC=YC+3
2610 RETURN
9990 SAVE "TAN▒"
9999 RUN
```


I
NIVEL
Y = RND
YC = RND
X = RND
XC = RND
S = 0
T = 100 -
-5*(N-5)
I = T TO 0
X, Y
XC, YC
X > XC
SIM
NÃO
XC-1
XC+1
C
C
NÃO
Y > YC
SIM
YC-1
YC+1
Y = YC
SIM
T
NÃO
INKEY$
SIM
K
NÃO
N


T
I = 62 TO 1
PLOT I, YC
EXPLOSÃO
S = S+1
SCORE
R


K
5
SIM
X−1
NÃO
6
SIM
Y+1
NÃO
7
SIM
Y−1
NÃO
8
SIM
X + 1
NÃO
9
SIM
A
R


A
I = 1 TO 62
PLOT I,Y
N
Y = YC
NÃO
SIM
R
EXPLOSÃO
S = S+1
SCORE



Renumerando

Ao se digitar um programa novo pela primeira vez devemos tomar o cuidado de deixar posições vazias entre o endereço de uma linha e o da linha seguinte. (Alguns programadores têm por habito numerar, por exemplo, de **10 em** 10). Isto é uma atitude prudente ditada pela Lei de Murphy: "Se algo tem chance de dar errado, certamente dará errado". Ao se rodar um programa recém-digitado, fatalmente descobre-se uma porção de "grilos", cuja eliminação exige a inserção de novas linhas. Após várias tentativas, o programa está pronto mas com uma numeração que fere o senso estético da maioria das pessoas. Além disso, um bom programador envergonha-se desta estranha numeração, pois sabe que outro pode deduzir os erros iniciais que ele cometeu pela numeração dos "remendos".

O programa **RENUMERANDO** visa eliminar estes inconvenientes. Inicialmente ele deve ser digitado e salvo em fita.

Antes de se digitar um novo programa ele deve ser carregado no computador. Na linha **9958** é conveniente colocar um monte de espaços vazios depois do título **RENUMERANDO**. Isto produz um hiato entre a rotina já inserida e o programa que você está digitando e ajuda a separação visual na listagem da tela.

Você deve ter o cuidado de digitar todos os endereços indicados por um **GOTO** ou **GOSUB** com quatro dígitos. Por exemplo, se no rascunho de seu programa você tem uma linha assim

```
312 GOTO 418
```

ela deve ser digitada assim

```
312 GOTO 0418
```

Uma vez digitado o rascunho do programa, basta comandar

```
GOTO 9959
```

que, após um certo tempo (30 segundos para um progrma de 1K), seu programa reaparecerá numerado a partir da linha **1000 de 10 em 10**.

Se algum **GOTO** ou **GOSUB** estiver fora das normas (4 dígitos), a rotina parará e sua linha **9972** indicará, na mensagem, em que linha do rascunho isto ocorreu.

A linha em questão deverá então ser corrigida e novamente a renumeração deverá ser reiniciada por um

```
GOTO 9959
```

Após um certo tempo de prática, o programador poderá dispensar a mensagem e substituir a linha 9972 por

```
9972 LIST L
```

Neste caso, ao detectar erro, a rotina já colocará o cursor de listagem ( ) na linha a ser corrigida, com economia de tempo.

Se o programa tiver algum **GOTO** ou **GOSUB**, obrigatoriamente fora de norma, por exemplo

```
312 GOTO (X+3)
```

haverá uma indesejada interrupção da rotina. Para driblar sua vigilância podemos usar a linha **9965** que pula os **REMs**. Neste caso inserimos um **REM** no rascunho:

```
312 REM GOTO (X+3)
```

que deverá ser oportunamente apagado após a renumeração.

Se quisermos alterar a numeração da linha inicial do nosso programa (mantendo-a porém com 4 dígitos) podemos mudar a linha

```
9976 LET B=1000
```

usando valores de **B** maiores que **1000**. Podemos também alterar a linha

```
9987 LET B=B+10
```

alterando com isto o incremento. Querendo esnobar em cima de um programa seguramente pronto e retocado, podemos numerar de 1 em 1:

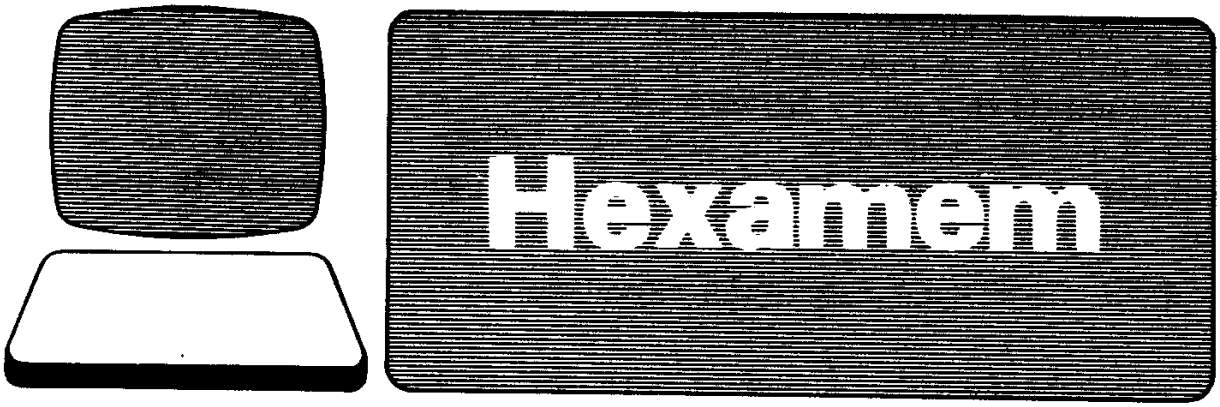
```
9987 LET B=B+1
```

Além disso, é conveniente certificar-se de que o computador esteja em **FAST** antes de se iniciar a rotina.

A única parte realmente enjoada disso tudo é finalmente ter de apagar as linhas da rotina uma vez terminada a tarefa. Se o programa, porém, não estiver absolutamente perfeito (lembre-se a Lei de Murphy!), e não houver limitação de memória, é conveniente deixar esta rotina pendurada no fim do seu programa ao salvá-lo em fita. Afinal de contas, nunca se sabe...!

```
9956 REM RENUMERANDO
9959 LET T$=""
9960 LET X1=16509
9961 LET X2=X1+2
9962 LET L=256*PEEK X1+PEEK (X1+
1)
9963 IF L>=9959 THEN GOTO 9976
9964 LET C=PEEK X2+256*PEEK (X2+
1)
9965 IF PEEK (X2+2)=234 THEN GOT
O 9969
9966 FOR I=X2+2 TO X2+C
9967 IF PEEK I=236 OR PEEK I=237
 THEN GOSUB 9971
9968 NEXT I
9969 LET X1=X2+C+2
9970 GOTO 9961
9971 IF PEEK (I+5)=126 THEN GOTO
 9974
9972 PRINT "COMANDO FORA DAS NOR
MAS          NA LINHA :";L
9973 STOP
9974 LET T$=T$+STR$ I+CHR$ PEEK
(I+1)+CHR$ PEEK (I+2)+CHR$ PEEK
(I+3)+CHR$ PEEK (I+4)
9975 RETURN
```

```
9976 LET B=1000
9977 LET X1=16509
9978 LET X2=X1+2
9979 LET L=256*PEEK X1+PEEK (X1+
1)
9980 IF L=9959 THEN STOP
9981 LET C=PEEK X2+256*PEEK (X2+
1)
9982 FOR I=1 TO LEN T$ STEP 9
9983 IF VAL T$(I+5 TO I+8)=L THE
N GOSUB 9990
9984 NEXT I
9985 POKE X1,INT (B/256)
9986 POKE (X1+1),B-256*INT (B/25
6)
9987 LET B=B+10
9988 LET X1=X2+C+2
9989 GOTO 9978
9990 FOR J=1 TO 4
9991 POKE (VAL T$(I TO I+4)+J),C
ODE (STR$ B)(J)
9992 NEXT J
9993 LET BYTE1=128+INT (LN B/LN
2+1)
9994 LET BYTE2=B*65536/(2**(BYTE
1-128))
9995 LET M=VAL T$(I TO I+4)
9996 POKE M+6,BYTE1
9997 POKE M+7,INT (BYTE2/256)-12
8
9998 POKE M+8,BYTE2-256*INT (BYT
E2/256)
9999 RETURN
```



O que é **Hexamem**? É um programa específico para se colocar programas em linguagem de máquina, a partir de um endereço fornecido pelo usuário.

Porém, é necessário reservar uma área da memória e para isto devemos modificar uma variável do programa interpretador: A **RAMTOP**. Esta serve para indicar ao computador até qual endereço a memória chega; ou seja, nos registros da memória correspondentes a essa variável, o programa interpretador coloca o endereço que seria do byte imediatamente após o último byte da memória.

A **RAMTOP** está colocada nos endereços **16388** e **16389**, pois, sendo ela um endereço, tem 16 bits devendo então ser "quebrada" em duas partes para poder ser armazenada, armazenando-se antes o byte menos significativo. Assim digite:

PRINT PEEK 16388 + (256 * PEEK 16389) (NEW LINE)

Você obtém o valor da **RAMTOP** que dependerá de quanta memória você tem disponível.

	RAMTOP	ÚLTIMO BYTE DA MEMÓRIA
2K	18432	18431
16K	32768	32767

O nosso programa já reserva a memória **30000** até **32767** e isto faz com que, ao se colocar um programa em Basic, suas variáveis e conteúdo da tela da TV **NUNCA** irão invadir esta região por "pensar" que a memória termina no endereço 29999. Esta região não será afetada pelo comando **NEW** e não pode ser passada para a fita através do comando **SAVE** a não ser em casos especiais.

OBS: Caso você não tenha expansão de memória use:

POKE 16388,173
POKE 16389,67
(reserva a região de **17325 a 18431**)

mude também a linha 30 por: **PRINT MEMÓRIA** (>=17325) retire as linhas 95,100 e de 200 a 580, substituindo a linha **105** por **PRINT TAB 13; AS (TO 2); TAB 17; AUX.**

Agora podemos seguir com o programa. Ao ser rodado, ele pede o endereço inicial de área a ser reservada (maior ou igual a 30.000).

Este programa fornece, também, uma "visualização" da memória do computador na tela da seguinte maneira:

MEM.	Nº da MEM. EM DE-CIMAL	CON-TEÚDO DA MEM. EM BI-NÁRIO	CON-TEÚDO EM HE-XADE-CIMAL	CON-TEÚDO EM DE-CIMAL

Assim você poderá entrar diretamente com os códigos hexadecimais. Para executar o que você introduziu em linguagem de máquina você deve dar "**XS**" (executa em **SLOW**) ou "**XF**" (em **FAST**) ou simplesmente **P** (para o programa sem executar nada).

```
   5 REM HEXAMEM
  10 FAST
  15 POKE 16388,48
  20 POKE 16389,117
  25 PRINT "ESCOLHA O ENDERECO I
NICIAL DA"
  30 PRINT "MEMORIA (>=30000)"
  35 INPUT IN
  40 PRINT
  45 PRINT "MEMORIA INICIAL= ";
  50 LET INI=IN
  55 LET A$=""
  60 SCROLL
  65 PRINT "MEM. ";IN;
  70 IF A$="" THEN INPUT A$
  75 IF A$="P" THEN STOP
  80 IF A$="X3" THEN GOTO 0130
  85 IF A$="XF" THEN GOTO 0135
  90 LET AUX=16*CODE A$+CODE A$(
2)-476
  95 LET X$=STR$ (16*CODE A$+COD
E A$(2)-476)
 100 GOSUB 0200
 105 PRINT TAB 13;C$;B$;TAB 24;A
$( TO 2);;TAB (31-LEN X$);X$
 110 POKE IN,AUX
 115 LET IN=IN+1
 120 LET A$=A$(3 TO )
 125 GOTO 60
 130 SLOW
 135 CLS
 140 PRINT AT 3,0;USR INI
 145 STOP
 200 LET I=1
 205 GOSUB 500
 210 LET I=2
 215 LET C$=B$
 220 GOSUB 500
 225 RETURN
 500 IF A$(I)="0" THEN LET B$="0
000"
 505 IF A$(I)="1" THEN LET B$="0
001"
 510 IF A$(I)="2" THEN LET B$="0
010"
 515 IF A$(I)="3" THEN LET B$="0
011"
 520 IF A$(I)="4" THEN LET B$="0
100"
 525 IF A$(I)="5" THEN LET B$="0
101"
 530 IF A$(I)="6" THEN LET B$="0
110"
 535 IF A$(I)="7" THEN LET B$="0
111"
 540 IF A$(I)="8" THEN LET B$="1
000"
```

```
 545 IF A$(I)="9" THEN LET B$="1
001"
 550 IF A$(I)="A" THEN LET B$="1
010"
 555 IF A$(I)="B" THEN LET B$="1
011"
 560 IF A$(I)="C" THEN LET B$="1
100"
 565 IF A$(I)="D" THEN LET B$="1
101"
 570 IF A$(I)="E" THEN LET B$="1
110"
 575 IF A$(I)="F" THEN LET B$="1
111"
 580 RETURN
 585 STOP
 600 SAVE "HEXAME"
 610 PRINT "DIGITE BREAK E LIST"
 620 PAUSE 300
 630 GOTO 600
```

APÊNDICE B

Tabela de Símbolos

Código
Caractere

Hexadecimal

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
	ESPAÇO											"	£	$	:	?
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	(	)	>	<	=	+	—	*	/	;	,	.	0	1	2	3
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Código
Caractere

Hexadecimal

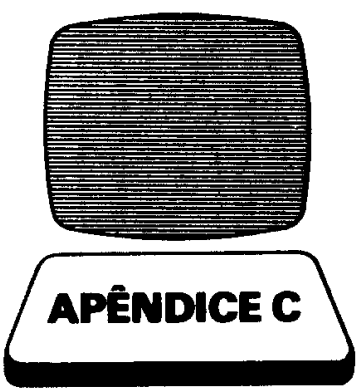
Hexadecimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
	RND	INKEY$	PI													
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
	CURSOR P/CIMA	CURSOR P/BAIXO	CURSOR P/ ESQ.	CURSOR P/DIREITA	GRAPHICS	EDIT	NEW LINE	RUBOUT	K / L MODO	FUNCTION					NÚMERO	CURSOR
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
													£	$	:	?
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
	(	)	>	<	=	+	—	*	/	;	,	.	0	1	2	3

Código
Caractere

Hexadecimal

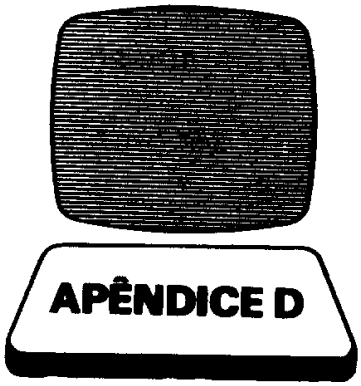
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
	""	AT	TAB		CODE	VAL	LEN	SIN	COS	TAN	ASN	ACS	ATN	LN	EXP	INT
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
	SQR	SGN	ABS	PEEK	USR	STR$	CHR$	NOT	**	OR	AND	<=	>=	<>	THEN	TO
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
	STEP	LPRINT	LLIST	STOP	SLOW	FAST	NEW	SCROLL	CONT	DIM	REM	FOR	GOTO	GOSUB	INPUT	LOAD
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
	LIST	LET	PAUSE	NEXT	POKE	PRINT	PLOT	RUN	SAVE	RAND	IF	CLS	UNPLOT	CLEAR	RETURN	COPY



Códigos de Erros

0 — Operação realizada satisfatoriamente.
Não ocorreu nenhum erro.
1 — Existe um NEXT sem nenhum FOR, mas existe uma variável com o mesmo nome.
2 — A variável utilizada não foi definida.
3 — Subscrito fora da faixa, mas se for usado um número maior que 65535 ou negativo ocorrerá erro B
4 — Toda memória já foi utilizada.
5 — Toda tela já foi utilizada. Utilize o comando CONT para continuar.
6 — Os cálculos utilizados superarão o número $1\phi^{38}$.
7 — RETURN sem GOSUB correspondente
8 — INPUT foi usado como comando direto
9 — A instrução STOP foi utilizada para interromper o programa.

A — Argumento inválido para certas funções.
B — Número inteiro fora da faixa.
C — O texto do argumento de VAL não forma uma expressão numérica válida.
D — O programa foi interrompido por BREAK
E — Não é usado.
F — O nome fornecido para o programa é uma string vazia.



APÊNDICE D

Vocabulário Basic TK

comando ou função	cursor	tecla	significado
ABS	F	G	Fornece o valor absoluto do número.
ACS	F	S	Transforma um número entre -1 e 1 em radianos (1 radiano -I- 57,3 graus).
AND	K ou L	SHIFT 2	Cria afirmações falsas ou verdadeiras para a tomada de decisões com o IF.
ASN	F	A	Transforma um número entre -1 e 1 em radianos.
AT	F	C	Usado com o comando PRINT para determinar a posição do caractere, string ou digito na tela. Precisa ser acompanhado de um endereço na forma, LINHA, COLUNA, onde a linha é um número entre 0 e 21 e a coluna, um número entre 0 e 31. Por exemplo: PRINT AT 8,10; "OLA"
ATN	F	D	Transforma um número entre -1 e 1 em radianos.
CHR$	F	U	Fornece o caractere correspondente a um código entre 0 e 255.
CLEAR	K	X	Apaga todas as variáveis e seus respectivos valores da memória, sem apagar o programa.
CLS	K	V	Apaga todas as informações da tela.

CODE	**F**	**I**	Fornece o código de um caractere, sendo este código um número entre 0 e 255.
CONT	**K**	**C**	Continuar a executar um programa que foi interrompido.
COPY	**K**	**Z**	Copia a tela.
COS	**F**	**W**	Fornece o cosseno de um número que representa um ângulo. O ângulo deve ser expresso em radianos (1 grau 0,175 radianos).
DIM	**K**	**D**	Define o nome, tipo, dimensão e o comprimento de uma array. Por exemplo: DIM N(10) Define N como uma array unidimensional, numérica com comprimento 10.
EXP	**F**	**X**	Fornece a exponencial de um número (exponencial de e 2,7182818).
FAST	**K ou L**	**SHIFT F**	Coloca o computador no modo FAST passando a operar com maior velocidade.
FOR	**K**	**F**	Define um loop, repetindo por um especificado número de vezes. Deve ser fornecido o valor inicial e final do lopp e o passo. Por exemplo: FOR I 1 TO 11 STEP 2 onde I é a variável de controle do loop que iniciará com o valor 2 e terminará quando a variável for 11 incrementando a variável de 2 unidades.
GOSUB	**K**	**H**	Transfere o programa para uma subrotina. O programa retorna à linha seguinte à linha com a instrução GOSUB quando o sistema encontrar o comando RETURN.
GOTO	**K**	**G**	Transfere o comando para a linha especificada. O número da linha pode ser expressado como uma

			variável numérica, valor de uma literal numérica ou por uma expressão numérica.
IF	**K**	**U**	Testa uma condição. Se for verdadeira executa o comando após a expressão THEN.
INKEY	**F**	**B**	Para cada instrução executada, o computador verifica se alguma tecla foi pressionada. Se alguma foi pressionada INKEY $ assume o valor da tecla pressionada, caso contrário, uma string vazia.
INPUT	**K**	**I**	Instrui o computador para ler um dado fornecido pelo teclado e armazena o dado em uma variável. Por exemplo: INPUT N$ lê a string introduzida pelo teclado e armazena-o na variável N$.
INT	F	R	Fornece a parte inteira de um número.
LEN	F	K	Fornece o comprimento de uma string.
LET	**K**	**L**	Especifica um valor a uma variável numérica ou alfanumérica. À esquerda do sinal igual "=" pode conter literal, variável ou expressão. Por exemplo: LET N = 3 LET N = M LET N = 5 * M LET A$ = "LET" LET A$ = B$ + C$ Qualquer variável citada à direita do sinal igual na instrução LET deve ser definida anteriormente.
LIST	**K**	**K**	Exibe na tela o programa. Se a instrução LIST for seguida por um número de linha, o programa listado será a partir desta linha. Por exemplo:

			LIST 200 a listagem começa a partir da linha 200. Caso contrário, se for dado o comando LIST, apenas, a listagem surgirá a partir da linha.
LN	**F**	**Z**	Fornece o logaritmo na base e de um número positivo.
LOAD	**K**	**J**	Carrega o programa gravado numa fita cassete. Este comando pode ser usado de duas formas: com o nome do programa: LOAD ("nome do programa") ou sem o nome do programa, apenas LOAD""
LPRINT	**K ou L**	**SHIFT S**	Idêntico ao PRINT, mas utiliza a impressora ao invés da tela. Não pode ser utilizada com a função AT, apenas com a função TAB.
LLIST	**K ou L**	**SHIFT G**	Lista na impressora o programa.
NEW	**K**	**A**	Limpa a memória do computador, apagando o programa e as variáveis da memória.
NEXT	**K**	**N**	Marca o fim do loop iniciado com a instrução FOR. A instrução NEXT precisa ser seguida pela variável determinada na instrução FOR. Por exemplo: NEXT I
NOT	**F**	**N**	Inverte o valor lógico de uma afirmativa.
OR	**K ou L**	**SHIFT W**	Cria uma combinação de sentenças verdadeiras e falsas, onde a instrução após THEN somente será executada se uma das sentenças for verdadeira. Caso contrário, executa a próxima instrução.
PAUSE	**K**	**M**	Instrui o computador para parar a

			execução do programa. O tempo de pausa depende do valor após a instrução PAUSE. Por exemplo: PAUSE 200 pára o programa por 4 segundos. (200/50 - 4 segundos)
PEEK	**F**	**O**	Fornece o conteúdo de uma memória entre 0 até 32767 (para computadores com 16 k).
PI	**F**	**M**	Valor de π (3,1415926...)
PLOT	**K**	**Q**	Exibe um "pixel" (um quarto de um caracetere). O sistema de endereçamento do PLOT está na ordem inversa à do PRINT AT; ou seja a posição 0,0 está no canto inferior esquerdo da tela. A forma geral do PLOT é: PLOT l,c onde l é a linha (0 a 43) e c é a coluna (0 a 63).
POKE	**K**	**O**	Introduz um valor numa memória localizada entre 0 e 65535. Deve ser utilizada na seguinte forma: POKE n,m onde n é o endereço da localização da memória e m é o valor a ser introduzido (decimal, inteiro entre -255 a 255).
PRINT	**K**	**P**	Imprime uma informação na tela. PRINT X;Y,Z - imprime na tela o conteúdo das variáveis X, Y, Z na mesma linha. O ";" não deixa espaços entre as informações. A "," faz com que a próxima informação o seja apresentada no próximo campo. A tela é dividida em 2 campos de 16 colunas cada. PRINT "mensagem" - imprime na tela a mensagem que está entre aspas.

			PRINT AT 1,c — imprime na tela a partir das coordenadas, pois 1 é a linha (entre 0 e 21) e c é a coluna (entre 0 e 32). PRINT TAB C — imprime na tela a partir da coluna C (entre 0 e 255).
RAND	**K**	**T**	Controla o início da seqüência de números randomicos produzidos pela função RND.
REM	**K**	**E**	É uma linha de comentária, ignorada durante a execução do programa.
RETURN	**K**	**Y**	Indica o fim de uma sub-rotina. O comando é transferido à linha seguinte ao GOSUB que o enviou a esta sub-rotina.
RND	**F**	**T**	Produz um número randomico entre maior ou igual a zero e menor do que 1.
RUN	**K**	**R**	Instrui o computador para começar a executar o programa. Se a instrução RUN estiver acompanhada pelo número de uma linha do programa, o computador começa a executar o programa a partir desta linha, caso contrário executa o comando CLEAR antes de iniciar a execução do programa.
SAVE	**K**	**S**	Grava o programa numa fita cassete. Este comando exige que o nome do programa seja colocado. SAVE "nome do programa"
SCROLL	**K**	**B**	Move uma linha para cima uma informação exibida na tela.
SGN	**F**	**F**	Fornece o sinal de um número resultando -1, 0 ou 1.
SYN	**F**	**Q**	Fornece o seno de um número que representa um ângulo que deve ser número em radianos.

SLOW	**K ou L**	**SHIFT D**	Retorna o computador no modo SLOW.
SQR	**F**	**H**	Fornece a raiz quadrada de um número não negativo.
STEP	**K ou L**	**SHIFT E**	Indica o incremento ou decremento na instrução FOR.
STOP	**K ou L**	**SHIFT A**	Para a execução do programa este comando pode ser introduzido ou não no programa.
STR$	**F**	**Y**	Transforma um número ou variável numérica em string.
TAB	**F**	**P**	Usada com a instrução PRINT ou LPRINT, imprimindo na tela ou na impressora a partir da coluna especificada.
TAN	**F**	**P**	Fornece a tangente de um número que representa um angulo em radianos.
THEN	**K ou L**	**SHIFT 3**	Usada com a instrução FOR.
TO	**K ou L**	**SHIFT 4**	Utilizada com a instrução FOR. É também usada para "slicing" strings. Por exemplo: LET N$ (3 TO 8) - "MAH"
UNPLOT	**K**	**W W**	Apaga a "pixel" da tela no endereço especificado. A forma geral é: UNPLOT c,1 onda c é a coluna (0 a 63) e 1 a linha (0 a 43). O endreço 0,0 na instrução UNPLOT está localizado no canto inferior esquerdo da tela.
USR	**F L**	**1**	Usada para chamar um sub-rotina em linguagem de máquina. Sua forma geral é: USR n onde n é o endereço em que começa a sub-rotina em linguagem de máquina.
VAL	**F**	**J**	Fornece o valor numérico de uma string.

Impresso nas oficinas da
EDITORA PARMA LTDA.
Fones: 66-3095 - 912-0790 - 912-0802 - 912-0819
Av. Antônio Bardela, 180
Guarulhos - São Paulo - Brasil
Com filmes fornecidos pelo Ed

FÁBIO RENDELUCCI
tem 18 anos e é professor de BASIC no Núcleo de Orientação de Estudos em São Paulo.
Graças à sua competência e juventude foi escolhido para lecionar numa turma experimental, composta de alunos adolescentes.
Todos eles já dominaram o BASIC mas queriam evoluir: optou-se por um curso de jogos para motivá-los, pois sua faixa etária estava entre 12 e 15 anos.
Com base nesta experiência didática foi escrito este livro, uma excelente continuação para quem já fez um curso de BASIC e quer se aprofundar divertindo-se.